

IEEE

MICRO

OCTOBER 1991

Chips, Systems, Software, and Applications

MOTION PERCEPTION

Also

- Computer analysis of myoelectric signals
- Programming distributed systems
- Performance and the i860
- FPGAs in ASICs

1951-1991



40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.



IEEE MICRO

Published by the IEEE Computer Society

October 1991

F E A T U R E S

8 Simulating a Function of Visual Peripheral Processes with an Analog VLSI Network

Hua Li and Ching-Ho Chen

Detecting motion with 2D hardware and algorithms

12 Computer Analysis of the Myoelectric Signal

Marco Knaflitz and Gabriella Balestra

The most promising computer applications of surface and needle myoelectric signal analysis

16 Alphorn: A Remote Procedure Call Environment for Fault-Tolerant, Heterogeneous, Distributed Systems

Hans-Ruedi Aschmann, Niklaus Giger, Elisabeth Hoeppli, Peter Janak, and Hubert Kirmann

Structuring large applications and communication over a variety of networks

20 A RISC Processor for Embedded Applications Within an ASIC

Charles E. Roberts

Handling the most functions with the least hardware, power, and cost

24 Performance and the i860 Microprocessor

Mark Atkins

Pipelining, parallelism, and internal caches help this million-transistor chip enhance performance

*Cover design: Chris Martin,
Design and Direction*

Cover photo: FPG International

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; IEEE Computer Society Headquarters, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription: \$21 in addition to IEEE Computer Society or any other IEEE society member dues; \$38 for members of other technical organizations. This journal is also available in microfiche form.

Postmaster: Send address changes and undelivered copies to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Second-class postage is paid at New York, NY, and at additional mailing offices.

Copyright and reprint permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US Copyright Law for private use of patrons those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress St., Salem, MA 01970. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to Permissions Editor, *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Copyright © 1991 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved.

DEPARTMENTS

- 2 From the Editor-in-Chief**
- 4 Micro Review**
Computers and creativity
- 6 Micro News**
- 28 Author's Guide**
- 30 On the Edge**
P1754, an open architecture
- 33 Micro Law**
Fraud on the Copyright Office
- 35 Software Report**
A glimpse of the future
- 36 New Products**
Systems, communications, DSP
hardware/software, test
equipment
- 42 Product Summary**
- C4 1992 Editorial Calendar**

*Call for papers, p. 41; Reader
Interest/Service/Subscription
cards, p. 64A; Advertiser/Product
Index, p. 80; Computer Society
information, cover 3*

IEEE Computer Society
PO Box 3014, Los Alamitos, CA 90720-1264
(714) 821-8380

Editorial: Unless otherwise stated, bylined articles and descriptions of products and services reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society. Send editorial correspondence to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. All submissions are subject to editing for style, clarity, and space considerations.

EDITOR-IN-CHIEF

Dante Del Corso
*Politecnico di Torino**

ASSOCIATE EDITOR-IN-CHIEF

Ashis Khan
*Mips Computer Systems, Inc.***

EDITORIAL BOARD

John Crawford
Intel Corporation

K.E. Grosspietsch
GMD

Joe Hootman
University of North Dakota

David K. Kahaner
*National Institute of Standards
and Technology*

Hubert D. Kirmann
Asea Brown Boveri Research Center

Priscilla Lu
AT&T

Richard Mateosian
Sandia National Laboratories

Nadine E. Miner
University of Tokyo

Ken Sakamura
University of Tokyo

John L. Schmalzel
University of Texas at San Antonio

Michael Slater
Microprocessor Report

John W. Steadman
University of Wyoming

Richard H. Stern
McDonnell Douglas Space Systems Co.

Philip Treleaven
University College London

Carl Warren
University of Manitoba

MAGAZINE ADVISORY COMMITTEE

James J. Farrell, III (chair)
Fiorenza Albert-Howard
Valdis Berzins
Jon T. Butler
B. Chandrasekaran
Carl Chang
Manuel D'Abreu
Dante Del Corso
Gerald Engel
Michael Evangelist
Sushil Jajodia
H.T. Seaborn
Pradip K. Srimani
Peter R. Wilson

STAFF

Marie English
Managing Editor

David Sims
Assistant Editor

H.T. Seaborn
Publisher

Marilyn Potes
Editorial Director

Douglas Combs
Assistant Publisher

Pat Paulsen
Assistant to the Publisher

Jay Simpson
Art Director

Joseph Daigle
Production

Christina Champion
Membership/Circulation Manager

Heidi Rex,
Marian Tibayan
Advertising Coordinators

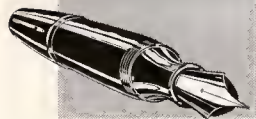
PUBLICATIONS BOARD

Ronald G. Hoelzeman (chair)
James Aylor
Victor R. Basili
Richard Burke
Jon T. Butler
J.T. Cain
B. Chandrasekaran
Carl Chang
Manuel D'Abreu
Dante Del Corso
Gerald Engel
Michael Evangelist
James J. Farrell, III
Tse-yun Feng
Anil K. Jain
Ted Lewis
Ray Miller
Michael C. Mulder
C.V. Ramamoorthy
H.T. Seaborn
Sallie Sheppard
Harold Stone
Earl E. Swartzlander
Peter R. Wilson

* Submit six copies of all articles and special-issue proposals to Dante Del Corso,
Dipartimento di Elettronica, Politecnico di Torino,
C.so Duca degli Abruzzi, 24, 10129 Torino, Italy; phone +39 11 564 4044;
Compmail: d.delcorso; Internet: delcorso@polito.it; Bitnet: delcorso@itopoli
or

** Ashis Khan, Mips Computer Systems, Inc., 950 DeGuigne Drive, Sunnyvale, CA 94086;
Internet: ashis@mips.com.

From the Editor-in-Chief



Bioengineering applications



BEFORE WE BEGIN to discuss the articles in this issue, I'd like to bring to your attention two items of news that will positively influence the coming issues of *IEEE Micro*. Priscilla Lu, from AT&T, starts her appointment with our editorial board this month; welcome! Priscilla's biography and photograph appear in *Micro News* this issue. In addition, we can now provide a more-detailed editorial plan for 1992. Please turn

to our back cover for this information.

Now let's turn to this issue, which was initially planned to be devoted to computer applications in biology and medicine and to bioengineering applications. We can offer a couple of articles related to this theme; they may open up new ideas for engineers and researchers, out of the classic scenery for microelectronics.

The first article, "Simulating a Function of Visual Peripheral Processes with an Analog VLSI Network" by H. Li and C. Chen, describes how one can design an integrated motion detector in a single piece of silicon. Motion detection is currently on the list of "hot" application problems. In this case note the design method, which relies on detailed mathematical formulation of the problem. The subject of this article inspired our front cover.

Then we go directly to the core of the issue theme. To pick up useful electrical signals from muscles, researchers have developed ad-hoc techniques that aim both at noise reduction and at limiting the discomfort for the sample under

measure (which is usually a person). These techniques, combined with technological developments, permitted electromyography techniques to move from research labs to current clinical practice. The second article, "Computer Analysis of the Myoelectric Signal" by M. Knaflitz and G. Balestra, is a structured overview of the problems and solutions that make up the area of work for biomedical engineers. A sequence of informational boxes builds a detailed tutorial introduction for those not familiar with handling biological signals.

After these tours near the boundary, we move to the remaining three articles, all of which are well inside the core of *Micro's* area of interest. These three articles ideally continue the flow of advanced processing architectures presented in the "Hot Chips" and "Far East" issues. "Alphorn, a Remote Procedure Call Environment for Fault-Tolerant, Heterogeneous, Distributed Systems" by H.R. Aschmann and others, presents a complete environment for programming, configuration, and communication in real-time distributed processing systems.

Next we dive deeply into hardware. As 2-million transistor chips become feasible, relatively small processors (which just a few years before could hardly be squeezed onto one chip) become library parts for the design of complex silicon systems. "A RISC Processor for Embedded Applications Within an ASIC" by C.E. Roberts is one further step in this direction. I would warn all readers: Watch *Micro's* pages carefully, and you will see an article in the near future something like "A Superscalar Processor as a Basic Cell for Super ASICs..."

The articles in this issue end with "Performance and the i860 Microprocessor" by M. Atkins. This article describes in detail the design solutions



that allow this processor to achieve such impressive performance announces the next processor of the family (one with 2.5M transistors). For an introductory tour of the i860 architecture and capabilities, you may want to

start with "Introducing the Intel i860" in the August 1989 issue of *Micro*.

Introducing the Intel i860

In the mailbag

(LK: liked, DLK: disliked, LTS: like to see)

October 1989

LK: Micro World, Micro Law; please keep them going; LTS: More on Europe, like Micro World.—N.B. (We will be able to keep these columns in our magazine, thanks to the good work of columnists Hubert Kirmann and Richard Stern.—D.D.C.)

February 1991

DLK: The total confusion over what article went where!!! Let's go back to a simple straightforward layout—each article in one place! I just got too lost.—R.N.C., Ryde, Australia (This problem still seems to be hot. Please continue sending your comments; anybody in favor of our current arrangement?—D.D.C.)

April 1991

LK: This is my first issue.... I liked it very much.—C.H., Helsinor, Denmark (And I am sure you will appreciate the next ones too!—D.D.C.)

DLK: Splitting articles—G.G., Fairfield, VA (See above.—D.D.C.)

LK: Micro Law, Micro View, New Products; DLK: Most articles are too

technical (mathematical). They belong in transactions! LTS: Any hope of a special issue on work in the developing countries?—M.T., Ikeja, Nigeria (Your first point is again credit for the respective columnists, Stern, Slater, and Hootman. Unfortunately, we had to discontinue Micro View for a while; but you will see it again in another form. Let me disagree with your second point: Our work demands something more than just a few mathematics, and the articles published in *Micro* substantially differ from the ones appearing in the various transactions. On your final point: Why not? All we need is a proposal and a guest editor. If any readers are interested, or know of someone who is, please contact either me or Ashis Kahn at the addresses listed on page 1 in this issue.—D.D.C.)

LK: Optical computations—A.N., Teheran, Iran

June 1991

DLK: Splitting articles between front and back of the magazine—S.N.K., Laurel, MO

Coming in December

Look for articles on database machines in the next issue of *IEEE Micro*

- VLSI accelerators for large database systems—from Bellcore
- An associative accelerator for large databases—from Laboratoire MASI in France
- RINDA: A relational database processor with hardware specialized for searching and sorting—from NTT in Japan
- A fine-grain architecture for relational database aggregation operations—from George Mason University
- A parallel, scalable, and microprocessor-based database computer for performance gains and capacity growth—from Naval Postgraduate School

Expedited delivery

is available to all members residing outside the USA, Canada, and Mexico. We invite you to take advantage of this service providing delivery of your magazine weeks earlier.



For information on this service and its cost, contact:

Expedited Delivery
IEEE Computer Society
 10662 Los Vaqueros Circle
 PO Box 3014
 Los Alamitos, CA 90720-1264
 USA
 Phone: (714) 821-8380
 Fax: (714) 821-4010

Micro Review

Richard Mateosian

2919 Forest Avenue

Berkeley, CA

94705-1310

(510) 540-7745

Computers and creativity

Last issue I reviewed *The Emperor's New Mind* by Roger Penrose, a book that is openly hostile to the view that computers can emulate people. The book that I've reviewed this time takes a contrasting position.

The Creative Mind—Myths and Mechanisms, Margaret A. Boden (Basic Books, a division of Harper Collins, New York, 1990, 316 pp., \$24.95)

Margaret Boden explores creativity from a scientific viewpoint. She looks at some impressive computer programs to see what insights they give us into the mechanisms of creativity. She also looks at myths and prejudices about using computers to understand or emulate human creativity.

Boden approaches creativity from the viewpoint of the computational psychologist. She is interested in the thought processes and mental structures that support creative behavior. Before she can begin, however, she must deal with two popularly held views that are antagonistic to any scientific approach to creativity. The inspirational view is like the picture of Mozart in the movie *Amadeus*, namely that creativity arises from a mysterious or divine spark that no amount of hard work or virtue can emulate. Thus the childish, boorish Mozart makes heavenly music while the hard-working Salieri's compositions never rise above the level of charming mediocrity. The romantic view is similar but less extreme. It holds that creative people have an innate talent—insight or intuition—that others lack. Some people are born creative, and the rest can only watch and admire them.

To Boden, inspiration and insight or intuition are not answers but merely ill-defined questions. She wants to know how people go about think-

ing new thoughts, and she looks to the field of artificial intelligence for ideas that will help us to understand this.

She poses and attempts to answer four questions:

- Can computational ideas help us to understand how human creativity is possible?
- Can computers ever do anything that appears to be creative?
- Can computers ever appear to recognize creativity, for example, in a poem written by a human?
- Can computers ever really be creative?

She calls these her four Lovelace questions, after Ada, Lady Lovelace, who first propounded the argument that many people would use in answering no to all of them: Computers cannot create, because they can only do what they are programmed to do. Boden answers yes to the first three Lovelace questions and probably not to the fourth, which she considers to be the least interesting for her purposes.

To proceed, Boden must first define creativity. She first reviews what creative people like the chemist Kekule and the poet Coleridge have said about the sources of their ideas. Kekule conceived the idea of the benzene ring while dozing in front of the fire and watching imaginary molecular strings and snakes gamboling in the flames. Coleridge devised *Kubla Khan* in a similar reverie. In the course of the book Boden returns to these examples and shows how mechanisms that Kekule and Coleridge did not report were actually at work in the forming of their creations.

In defining creativity Boden distinguishes psy-

chological creativity (P-creativity) from historical creativity (H-creativity). If Mary Smith has an idea that she could not have had before, and she recognizes its significance, the idea is P-creative. If no one has ever had that idea before, it is also H-creative. For the purpose of scientific analysis, Boden considers only P-creativity.

Having defined creativity, Boden begins to look at its mechanisms. She postulates that people build mental maps of conceptual spaces (for example, quantum theory in physics or impressionism in art). One mechanism of creativity is to apply some heuristic (for example, consider the negative) to change or extend the map. Thus Kekule's introduction of ring structure extended the conceptual space, which up to Kekule's time only included linear strings of carbon atoms. Boden sees AI as an ideal discipline for representing and exploring conceptual spaces.

Some of the techniques of AI are generative systems and heuristics. Generative systems allow systematic enumeration of points of the conceptual space. Heuristics are a priori considerations that allow searches of the conceptual space to be restricted to promising candidates. For example, a program trying to determine the home key of a fugue can use the very first note to suggest the four most likely candidates; one of these four will be the home key in almost all cases.

Representation is important in human thinking and in computer programs. The change from the roman to the arabic numbering system made possible huge advances in computation and ultimately in mathematics. Scripts, frames, and semantic nets are representations of knowledge commonly used in AI programs. Representations can assist creativity, but they can also inhibit it. For example, drawing a diagram can make it difficult to solve the following easy problem:

There are two houses x feet apart. A 20-foot string is sus-

pended between them. The points, A and B , at which the string is attached are at an equal distance above the ground. This distance is high enough to allow the string to hang freely without touching the ground. The vertical sag in the string is 10 feet. What is the value of x ?

No book dealing with current AI techniques can ignore neural nets. Boden shows how parallel distributed processing can be used to construct self-organizing systems that learn to recognize patterns and associate ideas in ways that are externally similar to what humans do. For example, she describes a system that learns the past tenses of English verbs, making the same pattern of errors that children make in the course of learning.

Boden considers some successful AI programs, because she believes that they shed light on the way human creativity works. For example, Harold Cohen, an established artist, has produced a program called Aaron, which produces drawings that many human artists would be proud of. Aaron contains general knowledge about composition and drawing techniques and specific knowledge about the kinds of subjects it draws. Its internal representations allow it to vary its drawings indefinitely within a set of well-defined constraints that define its "style."

Boden also considers a jazz improvisation program that models the styles of Charlie Parker and Dizzy Gillespie. Noting that jazz pianists make up these improvisations as they go along, the programmer has found algorithms that require little computing or memory to execute. The programmer claims that the program performs as well as a moderately competent beginner.

Even more interesting to me is a program called Acme, which recognizes analogies. Acme can understand and explain Socrates' characterization of himself as a midwife of ideas. Acme

uses a connectionist system called Word Net as its dictionary. The entries in Word Net are concepts, and the links between them are relationships like synonym, antonym, subordinate, and part.

In addition to the above examples of artificial intelligence programs emulating artistic creativity, Boden also gives numerous examples of programs in the scientific field. She describes expert systems like Dendral, the well-known molecular analysis program, and a program that successfully diagnoses soybean diseases. These programs use induction to revise and improve their decision making. Thus, the soybean program diagnosed correctly only 83 percent of cases using its initial set of rules, but after learning from a large set of examples it improved its percentage to nearly 100 percent.

Abstract Mathematician is a program that starts with 100 primitive mathematical concepts and about 300 heuristics and looks for interesting patterns. This program has discovered several new theorems and a variety of well-known ideas like the famous Goldbach conjecture (every even number greater than 2 is the sum of two different primes).

Having led the reader through a number of quite impressive examples, Boden returns to the Lovelace questions and explains her answers to them. To do this, she must first tear down another straw man. To counter the argument that unpredictability is the essence of creativity, she takes apart and analyzes the concepts of chance, chaos, and randomness. Then she points out that prediction is not what science is all about, but only a side effect of its real goal: To explain how it is possible for things to happen as they do.

One of the most important points that Boden makes in her demystification of creativity is that its mechanisms are available to everyone. She first asks the reader to recount all the thoughts that occur during an attempt to com-

plete a limerick with a given first line. Then she says to try the exercise again, thinking out loud. In the second case, the mental processes are much more apparent. The point is that we don't normally remember all of the things that go on inside our heads, and neither do people like Coleridge or Kekule who report on the mystical ways that ideas come to them. In Boden's view creativity entails abilities that we all have: noticing, remembering, seeing, speaking, hearing, understanding language, understanding analogies. Mozart may have been better at some of these things than the rest of us, but he belonged to the same genus and species.

The contention that creativity proceeds by mechanisms available to all of us is essential to Boden's belief in her positive answers to the Lovelace questions. To her, the rejection of machines as potential advisers, companions, entertainers, and so on smacks of human chauvinism rather than a provable difference between what we do and what they might someday be capable of doing.

I picked up this book, and I read it every available moment until I finished it. Margaret Boden writes creatively about creativity. She writes about mental associations and allusions in a style filled with associations and allusions. Time and again she reintroduces old examples in surprising new contexts or simply turns a very clever phrase.

Boden takes examples from many fields of art and science, and she seems at home in all of them. In one case, for example, having referred much earlier to Shakespearean sonnets, she is talking about programs that can find rhymes. She mentions that the program might even be able to see love/prove as a rhyme. This statement stands by itself, but some readers will realize that it is an allusion to the fact that no less than four of Shakespeare's sonnets use that rhyme in their closing couplets.

I strongly recommend this book to anyone interested in creativity and the field of artificial intelligence research.



Send information for inclusion in Micro News one month before cover date to Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

Organic compounds: Future electrical materials?

New, lower priced batteries, semiconductors, and other electrical devices capable of being produced at much lower temperatures may one day prove possible if current materials research succeeds.

Chemists at last February's meeting of the American Association for the Advancement of Science reported findings of interesting electrical, optical, or magnetic properties in a series of synthesized compounds. These one-dimensional materials also differ from metals in resistance and properties. One salt known as TTeF-TcNQ uses tellurium to improve its electrical properties. This material remains metallic to temperatures as low as 1.5 Kelvin.

Dwaine O. Cowan, Theodore Poehler, and Thomas Kistenmacher of Johns Hopkins University have demonstrated the possibility of constructing organic solids that exhibit either metallic or superconducting properties under the proper conditions. Though the compounds are difficult to make, several groups in the US, France, Japan, and Denmark report efforts designed to exploit the potential benefits of the compounds.

Cowan's group created the first organic metal in 1972, and former Hopkins postdoctoral fellow Klaus Bechgaard created the first organic superconductor 10 years ago.

Beyond the stethoscope

"Sounds made by normal and defective hearts frequently differ so subtly that only a highly trained heart specialist with years of experience can tell the difference." Though this statement from Yale University physicist William R. Bennett, Jr., may sound obvious, it pinpoints the reasons Bennett's diagnostic invention has proved so helpful.

Though heart defects produce distinctive sounds, most physicians find them hard to recognize because they may see only a few patients in a lifetime with certain heart defects. Bennett's dynamic spectral phonocardiograph helps physicians by greatly amplifying sound differences; it is much more sensitive than a stethoscope and works successfully in noisy hospital environments.

Using a Macintosh IIci, Bennett created a software program that first converts heart sounds into numbers and then into images via a fast Fourier transform. As shown in the figure, a print-out of the sounds displays loud beats as jagged peaks and softer murmurs as shallow squiggles. A color monitor lets users distinguish red normal heart sounds from yellow abnormal sounds. "The phonocardiograph will enable physicians to watch the computer image of the heartbeat at the same time they are listening to the sound," says Bennett, who is the C. Baldwin Saw-

yer Professor of Engineering and Applied Science at Yale.

Bennett hopes his invention will help physicians avoid unnecessary surgery or pick the best time for surgery. Reeves Scientific Inc. of New Haven builds prototypes of the phonocardiograph for testing in 52 US teaching hospitals.

In addition to the heart monitor, Bennett's interest in sound waves has also spawned a design for a hearing aid that uses a chip to provide more accurate sound and less background noise than existing hearing aids.

Though still under development, his hearing aid was patented last year.

In 1960 while at Bell Laboratories, Bennett coinvented the first gas laser, and in 1964 as a Yale faculty member, he invented the argon ion laser.

The challenge of open buses

Why do we need open buses? Attendees at the Open Bus Systems 91 conference in Paris this November 26-27th will hear Paul Borrill answer that very question. Other talks include David Wilner discussing new tech-

niques for improving reliability of complex life-critical computer systems, and Cristopher Eck reporting on the VMEbus in the 90s. David Gustavson's invited talk will focus on the Scalable Coherent Interface and new, related standards projects. Session topics include industrial applications, multiprocessor architectures, Futurebus+, real-time software, conformance testing, image processing, SCI, and board design.

For information, write to Open Bus Systems 91, VITA Europe, PO Box 192, NL-5300 AD Zaltbommel, The Netherlands; telephone at +31 4180 14661 or fax to +31 4180 15115.

Editorial board expands

Editor-in-Chief Dante Del Corso has appointed Priscilla Lu, director of imaging systems at AT&T Computer Systems, to the editorial board of *IEEE Micro*. Lu will review manuscripts for the magazine.

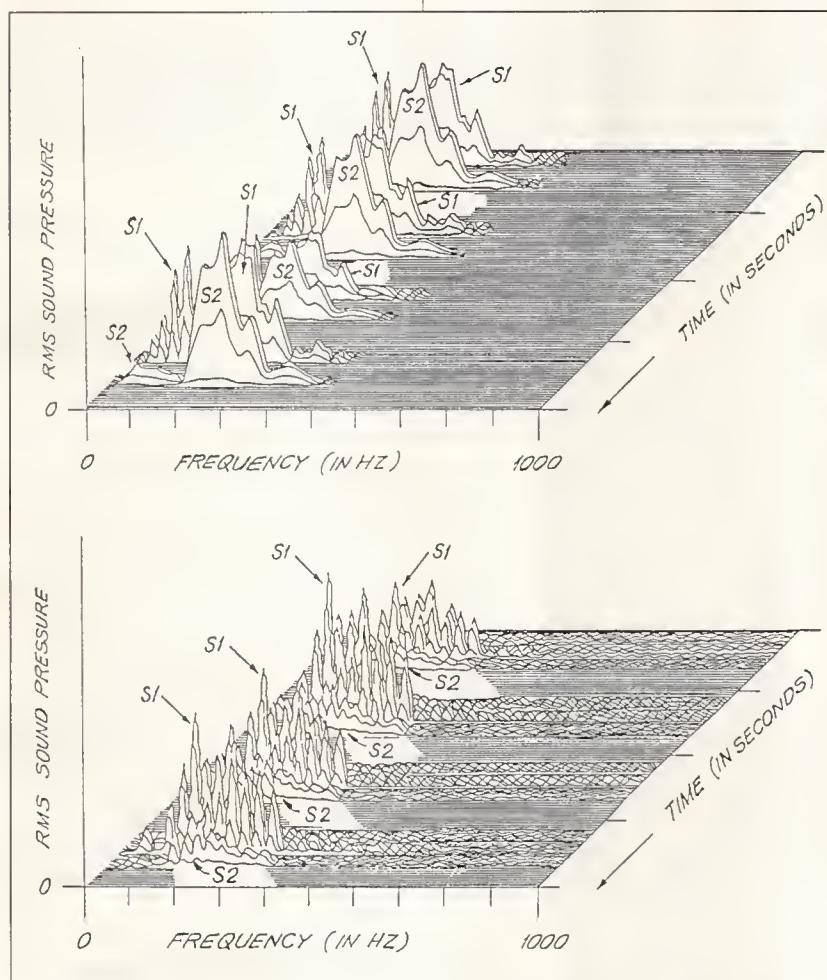
Lu joined AT&T as first supervisor of the microprocessor architecture group, became department head of the Microsystems Engineering Department, and later moved to the position of department head of the Network and Systems Management organization for the Star Group product line.

Lu holds a PhD in computer science and electrical engineering from Northwestern University and an MS and BS in computer science and mathematics from the University of Wisconsin, Madison.

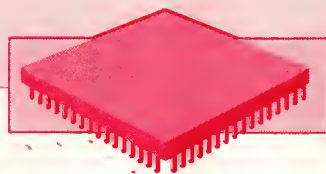
Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 198 Medium 199 High 200



Phonocardiogram printout of a normal heartbeat producing two sounds, S1 and S2, followed by a silence represented as straight lines (top); and an abnormal heartbeat producing a murmur displayed as wavy lines (bottom).



Simulating a Function of Visual Peripheral Processes with an Analog VLSI Network

Using analog VLSI technology to implement algorithms that mimic human visual perception opens up a new avenue to approach machine intelligence. Our 2D network simulates a function of motion perception in the visual peripheral process, carrying out computations with analog voltage and current. SPICE simulations and experiments using real images matched the results of the theoretical analysis.

Hua Li

Ching-Ho Chen

Texas Tech University

General experience leads us to believe that the human visual perception of motion consists of two processes: peripheral, which detects motion at a low level, and attentive, which interprets the motion.^{1,2} The peripheral process primarily directs the focus of attention. It is important in guiding one's reaction, orienting responses and locomotion, and usually serves as the first stage for cognition. For example, we first detect moving objects and record their rough locations and sizes. Then, we decide whether further high-level processing is necessary.³ The peripheral process is presumably spatially parallel so that the entire visual field is processed quickly.

Computationally, we can model the peripheral process by noting the differences between two consecutive images. Jain recorded some interesting work on the mathematical formulation of the process.⁴ He developed a *difference-picture technique* for dynamic image processing. The crucial part of simulating the peripheral process is its processing speed. It must be fast enough to allow an intelligent machine or robot to interact with the instantaneously changing environment.

Using analog VLSI (very large scale integration) technology to implement algorithms that mimic human visual perception gives us new insights into machine intelligence. Mahowald and Mead have constructed a silicon retina.⁵ Hutchinson, Koch, Luo, and Mead have built a

resistive network to compute optical flow.⁶ Lumsdaine, Wyatt, and Elfadel have worked on nonlinear analog networks for image smoothing and segmentation.⁷ Recent reports indicate that some silicon models have been suggested to simulate human eye saccadic movement⁸ and vertebrate retinal processing.⁹ One of the common features of these works is that expensive numerical computations are performed in an analog way with very fast speeds.

We describe a two-dimensional analog VLSI network for implementing a function of motion perception during the peripheral process. The network contains many nodes, or processing elements (PEs), which operate concurrently in asynchronous fashion. Each PE performs an image-difference operation as well as thresholding to extract motion information on a pixel-by-pixel basis. Analog voltage, which interacts extremely fast, implements all computations. An energy equilibrium state of the network provides stabilized analog voltage across the network—the solution to the problem. The structure of each simple PE has nice regularity.

The difference-picture technique

Implementing vision to build an intelligent machine is a most exciting and difficult task. Human beings perceive environment effortlessly in comparison to today's sophisticated computer vision systems. Consequently, we should take this

suggestion from nature and find an alternative to design image processing algorithms. Many research works have explored the feasibility of developing computational models to imitate human visual perception.¹⁰⁻¹² It is well known in visual sciences that early motion perception uses a mechanism that can be described as a delayed comparison scheme.¹³ The difference-picture technique relies on this scheme. With some modifications, researchers have recently applied the technique to design motion-compensation algorithms and circuits for applications in the high-definition television industry.¹⁴

Given an image $E(x, y, t)$ as a function of both location (x, y) and time t , we assume partial derivatives $[\partial E(x, y, t) / \partial x, \partial E(x, y, t) / \partial y, \partial E(x, y, t) / \partial t]$ exist. The $[\partial E(x, y, t) / \partial x]$ and $[\partial E(x, y, t) / \partial y]$, generally speaking, are related to the image edge segments, while $[\partial E(x, y, t) / \partial t]$ relates to time. We calculate $[\partial E(x, y, t) / \partial t]$ as follows:

$$\frac{\partial E(x, y, t)}{\partial t} \equiv \frac{E(x, y, t - \Delta t) - E(x, y, t)}{\Delta t} \quad (1)$$

where $x = 0, 1, \dots, M$ and $y = 0, 1, \dots, N$ for $M \times N$ images with Δt equaling one time unit or sampling interval. For nonstationary images, we can use Equation 1 to define secondary parameters such as $D(x, y, t)$, $D_1(x, y, t)$, and $D_2(x, y, t)$ to describe motion.

Definition 1. We define a difference picture $D(x, y, t)$ as a binary image. Depending on the threshold T of the image difference $E(x, y, t) - E(x, y, t - dt)$, each of its pixels has two possible gray levels L_l and L_b . That is

$$D(x, y, t) = \begin{cases} L_b, & \text{if } E(x, y, t) - E(x, y, t - dt) \geq T; \\ L_l, & \text{otherwise.} \end{cases} \quad (2)$$

Here $L_b = 255$ and $L_l = 0$. T is chosen heuristically and is set high whenever the images are badly corrupted by noise. We obtain $D(x, y, t)$ based on the pixel-by-pixel determination of significant changes of image intensity, with its nonzero regions corresponding to the nonstationary scene.

We further decompose a difference picture $D(x, y, t)$ into two subsets $D_1(x, y, t)$, $D_2(x, y, t)$ that correspond to nonstationary regions of images $E(x, y, t)$ and $E(x, y, t - dt)$.

Definition 2. We define subdifference pictures $D_1(x, y, t)$ and $D_2(x, y, t)$ as subsets of $D(x, y, t)$, such that $D_1(x, y, t) \subseteq E(x, y, t)$, $D_2(x, y, t) \subseteq E(x, y, t - dt)$, and

$$D(x, y, t) = D_1(x, y, t) \cup D_2(x, y, t), \quad (3)$$

as well as

$$D_1(x, y, t) \cap D_2(x, y, t - dt) = \phi. \quad (4)$$

The relative position between $D_1(x, y, t)$ and $D_2(x, y, t)$ describes the *perceived* motion direction, the size of each

nonstationary region, as well as its location. With this information, higher level processes of cognition can be issued. Computationally, a fast and accurate algorithm simulating a peripheral process can serve as a preprocessing stage, which can reduce the search space and improve computational speed and efficiency.^{3,4}

Design methodology

We can realize the difference-picture technique by a 2D analog VLSI network. Such hardware implementation provides very fast computational speed. This network contains $M \times N$ nodes, or PEs, with each corresponding to one image pixel. Each PE has two functions: computation of image difference $E(x, y, t) - E(x, y, t - dt)$ and thresholding the difference to derive a difference picture $D(x, y, t)$. We can alter the overall network either through programming or electronically. This capability allows us to work with a large class of motion-detection problems.

Modifiable thresholding unit. To perform analog computing, we first map each gray level of image intensity to analog voltage.

Definition 3. An analog voltage V_i , such that $\{V_i \mid 0 \leq V_i \leq V_{dd}, i = 0, 1, \dots, 255\}$, satisfying the equation of the form

$$V_i = \frac{V_{dd}}{L_b - L_l} (L_i - L_l), \quad (5)$$

is said to be a mapping voltage, where L_b is the largest gray level of image intensity and L_l is the smallest one.

The difference-picture technique gives T in terms of the gray level of image intensity which is to be converted to analog voltage to provide a specification for hardware design. In the following, unless otherwise stated, we refer to a *threshold voltage* as a mapping voltage of T . The following theorem describes the mapping.

Theorem 1. If we choose T as a gray-level threshold for a given image and if we use a mapping in Definition 3, the analog threshold voltage V_{in} equals

$$V_{in} = \frac{V_{dd} T}{L_{max}}, \quad (6)$$

where $0 \leq T \leq 255$, L_{max} is the maximum gray level of image intensity. In our case it equals 255. V_{dd} is a standard 5V power supply.

Proof. Following Definition 3, let

$$V_{in} = V_i, \quad (7)$$

$$L_{max} = L_b - L_l, \quad (8)$$

$$T = L_i - L_l. \quad (9)$$

Then, substituting them into Equation 5 gives us Equation 6.

With this simple mapping, we can now interpret Equation 2 as one *voltage transfer characteristic* (VTC) curve with a turning point at V_{in} . This VTC has low output if the input is low and has high output if its input is high.

Definition 4. A point on a VTC curve is said to be a threshold voltage V_{in} , if it is the point satisfying

$$V_{in} = \frac{1}{2}(V_{max} - V_{min}), \quad (10)$$

where V_{max} and V_{min} are the maximum and minimum voltages of the VTC.

Figure 1a illustrates the transition of the VTC with a threshold V_{in} . It is not difficult using two stages of inverters to realize the VTC. We selected the CMOS (complementary metal-oxide semiconductor) technology because it consumed low amounts of power. In addition, it produced very little distortion in terms of the input-output voltage range that is desirable since the input images ranging from zero to 255 gray levels should have the same gray-level range for the output image.

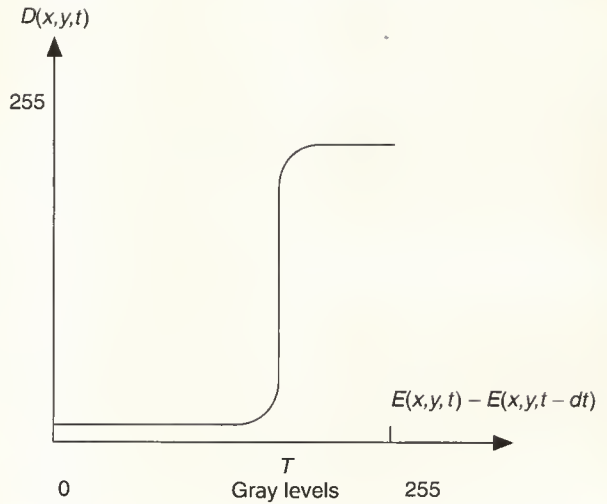
Figure 1b shows the circuit configuration. Note that we designed the circuit based on quite different performance considerations than a digital case. Digital circuit design requires a fixed threshold and fixed VTC. However, they are not acceptable here, since the motion-detection scheme must have a modifiable threshold and modifiable VTC to handle different motion situations and to eliminate random image noise. In addition, digital circuit design emphasizes large noise margins, which are not the concern in designing our thresholding unit.

One of our design objectives is to implement threshold voltage V_{in} according to different T 's. From the viewpoint of hardware implementation, we can relate the threshold voltage V_{in} to a W/L ratio between the PMOS load and the NMOS driver. The following theorem gives the design principle.

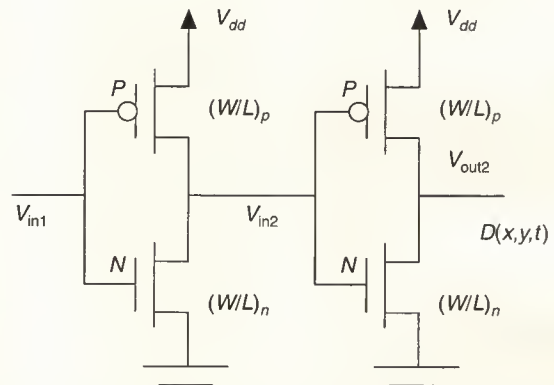
Theorem 2. For a given arbitrary threshold $\{T \mid L_i \leq T \leq L_b\}$, we realize a voltage threshold V_{in} defined in Theorem 1 by defining the W/L ratio using the following form of the equation,

$$\sqrt{\frac{\left(\frac{W_n}{L_n}\right)}{\left(\frac{W_p}{L_p}\right)}} = \frac{V_{dd} - V_{in} - |V_{TP}|}{V_{in} - V_{TN}}, \quad (11)$$

where W_n and W_p indicate the channel widths of the NMOS and PMOS devices, and L_n and L_p represent the corresponding channel lengths. V_{TN} and V_{TP} are the device threshold voltage of NMOS and PMOS transistors. We prove Theorem 2 as follows.



(a)



(b)

Figure 1. Curve describing Equation 2, which is to be implemented by an analog modifiable thresholding unit (a); and CMOS circuit designed to achieve the VTC (b). P and N indicate PMOS and NMOS.

Proof. Considering a VTC with a threshold V_{in} , since both PMOS and NMOS are in saturation mode during the transition, we have $I_{DN(sat.)} = I_{DP(sat.)}$. That is

$$\frac{K_n}{2}(V_{GSN} - V_{TN})^2 = \frac{K_p}{2}(V_{SGP} - |V_{TP}|)^2 \quad (12)$$

where $V_{GSN} > 0$, $V_{TN} > 0$, and $K_n = K(W_n/L_n)$, $K_p = K(W_p/L_p)$.¹⁵ From the circuit configuration in Figure 1b, we have

$$V_{in} = V_{GSN}, \text{ and } V_{SGP} = V_{dd} - V_{in}. \quad (13)$$

Substituting these into Equation 12, gives us

$$\frac{W_n}{L_n} (V_{in} - V_{TN})^2 = \frac{W_p}{L_p} (V_{dd} - V_{in} - |V_{TP}|)^2 \quad (14)$$

After some mathematical manipulation we get the final form given in Equation 11.

Obviously, we need the cascaded CMOS inverters so we can implement the transfer function given by Equation 2.

Lemma. For cascaded two-stage cases, the W/L ratio defined in Equation 11 still holds.

Proof. We provide an instructive proof of the lemma. Denote V_{in1} , V_{in2} , V_{o1} , and V_{o2} as inputs and outputs of the first stage and the second stage, where $V_{o1} = V_{in2}$. With this relation, transistors at two stages can both reach a saturation mode during time period Δt . Hence the equation of current $I_{DN(sat)} = I_{DP(sat)}$ holds for the second stage. Therefore follow-

ing the same argument in the proof of Theorem 1, we derive Equation 11 to describe the relationship between threshold V_{in} and the W/L ratio.

We can choose a W/L ratio to get a different T for motion-detection applications. We can modify the W/L ratio by burning fuses. For example, we can fuse a wire with 1-by-6 micrometer dimension on a 0.1-micrometer resistive layer.¹⁶

Analog differencing unit. We realize the differencing operation defined in Equation 2 by using a high-gain, large-swing, CMOS buffer amplifier suggested by Nagaraj¹⁷ to build a noninverting opamp configuration. We connect two input nodes V_i and V_j to image $E(x, y, t)$, $E(x, y, t - dt)$ at pixel location (x, y) and then connect the output node to the thresholding unit. This configuration appears in Figure 2a, which defines one node, or called PE, of the network. With each of these nodes, we can then construct the 2D analog

continued on p. 44

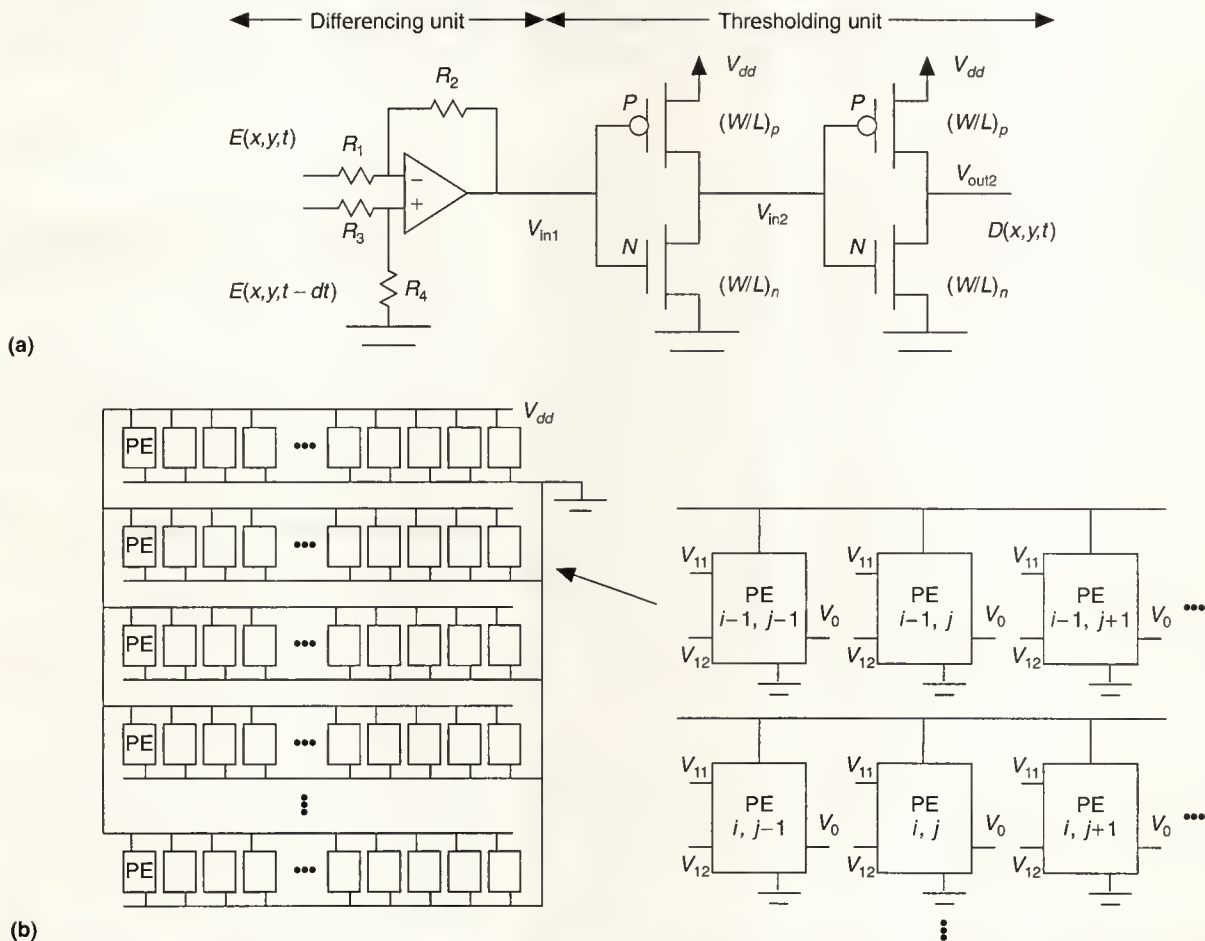


Figure 2. The noninverting configuration that performs differencing operation of two input images is used as the first stage of the PE (a). A schematic diagram of the 2D analog VLSI network (b).



Computer Analysis of the Myoelectric Signal

Powerful and relatively inexpensive computer systems allow clinicians to use signal processing techniques previously applied only in basic research. Biomedical engineers have adjusted research electromyography techniques to match physicians' needs. In clinical analysis of the myoelectric signal detected with needle electrodes and noninvasive probes, computers are crucial.

Marco Knaflitz

Gabriella Balestra

Politecnico di Torino

During the last 50 years, physicians have been able to analyze myoelectric signals collected by needle electrodes inserted into muscles. As a result, they have gained valuable knowledge about muscle structure and function, and about pathological processes of muscles and the central and peripheral nervous systems. (For earlier works, see the accompanying box.) Although myoelectric signal analysis was worthwhile in clinical practice, its clinical relevance was definitely more limited than that of electrocardiography: The signal produced by the heart is much easier to interpret because it is quasideterministic and periodic. With simple amplifiers and monitors, cardiologists can extract a large amount of information about the heart function by studying signal morphology in the time domain. Such a simple approach is not satisfying with the myoelectric signal.

Since the late 1960s, the availability of computer systems and software for digital signal processing has encouraged the development of new techniques for analyzing the myoelectric signal. Researchers focused on two main subjects:

- the relationships between the myoelectric signal collected by surface electrodes and the working of muscles and the nervous system, and
- new techniques to extract information about

the central nervous system's strategy for controlling motor units.

Approximately 15 years of basic research and the availability of powerful and relatively inexpensive computer systems have resulted in the development of several new techniques in myoelectric signal analysis. These techniques are finding ready application in standard clinical practice. We review the most promising applications of computers to the analysis of both surface and needle myoelectric signals.

Basics of myoelectric signals

A skeletal muscle comprises a number of motor units. Each motor unit consists of muscle fibers innervated by the terminal branches of a single α -motoneuron whose cell body is located in the anterior horn of the spinal cord. The motor unit is the smallest part of a muscle that the central nervous system can control individually. The central nervous system controls muscle force by adjusting the number of recruited motoneurons and the firing frequency of each motoneuron. These two factors are determined by the synaptic activity on the cell body of each motoneuron, which, in turn, is a function of the activity of the descending upper motoneurons and the afferent activity arriving from peripheral sensors.

Figure 1 schematically represents myoelectric

Early work with muscle-generated electricity

In 1666, Italian scientist Francesco Redi inferred that muscles generate electricity, and in 1791 Luigi Galvani observed the effect of electric current on muscle contraction. Using the first galvanometers, Carlo Matteucci demonstrated in 1838 that electric currents originate in muscles during muscle contraction.

Studies of the electrical activity of muscles were frequent in the 19th century, and in 1912 H. Piper reported the link between myoelectric signal spectral characteristics and muscle fatigue. In 1922, Herbert S. Gasser and Joseph Erlanger used the first cathode-ray oscilloscopes to observe the morphology of the myoelectric signal. Their observations on the electrical activity of muscles earned them a Nobel prize in 1944.

Because of the stochastic nature of the myoelectric signal, only rough information could be obtained from its observation until 1960. Initially, the myoelectric signal was used as a sign of muscle contraction—for example, in studying the activation pattern of lower limb muscles during locomotion. In such applications, extracting the important information from the signal is easy, because it is simply represented by the presence or absence of the signal itself. Since 1930, devices to detect, amplify, and display the time course of the myoelectric signal have been available.

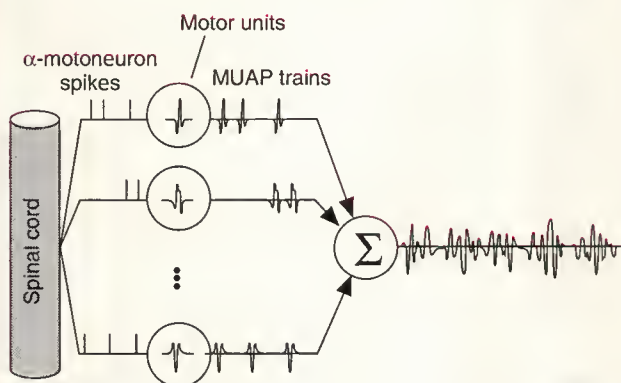


Figure 1. Schematic representation of the myoelectric signal generation process.

signal generation. We can describe the firing pattern of an active motoneuron as a train of δ functions and the distance between consecutive δ pulses as the interspike interval, which is a random variable. The instantaneous firing rate is the inverse of the interspike interval. The average firing rate of a motoneuron and its standard deviation depend on the synaptic input to that neuron. Firing rates of different α -motoneurons are, in general, different and uncorrelated.

Each terminal branch of a motoneuron innervates a muscle fiber in an intermediate point. Through the neuromuscular junction, the firing of a motoneuron triggers two depolarization zones that propagate in opposite directions toward the two ends of the muscle fiber, generating two action potentials traveling at a speed called the muscle-fiber conduction velocity.

The sum of the contributions of the fibers belonging to a particular motor unit is the motor-unit action potential (MUAP), and the sequence of these motor-unit action potentials is the MUAP train.

The myoelectric signal detected at the surface of the skin or inside the muscle is the summation of the contributions of individual MUAP trains. Because motor-unit discharges are irregular and MUAPs have different shapes, we can think of the myoelectric signal as a band-limited stochastic process with a Gaussian amplitude distribution. Its frequency spectrum ranges from DC to approximately 500 Hz.

Before we can store and observe it, the tissue, skin-electrode interface, electrode configuration, and recording apparatus subject the surface myoelectric signal to a series of filtering effects. Thus, the observable myoelectric signal is a function of anatomical and physiological factors, which give the signal its intrinsic characteristics, and the filtering features of the environment and the apparatus, which give it extrinsic characteristics. The methodology used to detect and record the signal can modify the extrinsic characteristics (see box on the next page). The structure and workings of the muscle determine the intrinsic characteristics.

Surface myoelectric signal analysis

We generally refer to the myoelectric signal collected with electrodes placed on the skin above a muscle as the *surface myoelectric signal*. SMES and the myoelectric signal detected by needle electrodes mainly differ in:

- The detection volume of surface electrodes is much larger than that of needle electrodes—several cubic centimeters compared with fractions of cubic millimeters. As a consequence, signals coming from several tens of motor units constitute the SMES, while needle electrodes detect a signal consisting of the activity of a few muscle fibers belonging to 10 to 15 motor units.
- Because of the larger distance of muscle fibers from surface probes, the SMES power spectrum is generally

Myoelectric signal detection and amplification

The term *electrode* describes a sensor that consists of two distinct parts: a detection surface, typically metallic, which comes in contact with the skin and senses the myoelectric signal, and the appropriate housing for the detection surface. The area of the detection surface affects the electrode's impedance and its detection volume. The detection volume is the space from which the electrode senses the signal—the larger the detection surface, the lower the impedance and the larger the detection volume.

Monopolar and bipolar detection techniques. Myoelectric signals can be detected by different surface electrodes placed on the skin above the muscle to be investigated. Figure A shows the two basic techniques, generally referred to as *monopolar* and *bipolar*.¹ In the monopolar configuration, only one detection surface is placed on the skin above the muscle to be investigated. This electrode detects the electric potential with respect to a reference electrode placed in an environment unaffected by the electrical activity generated by the muscle to be studied.

The bipolar detection configuration yields increased spatial resolution and improved noise rejection. This configuration uses two detection surfaces to sense the voltage potential at two locations on the skin with respect to a reference electrode. The two signals sensed at the detection surfaces feed into a differential amplifier, whose output generates the single-differential EMG signal.

Active electrodes. When relatively large common-mode signals are generated on a patient, a major problem is the generation of a differential mode signal resulting from common-mode excitation in the presence of an imbalance of the electrode impedances. This situation is rather common when detecting bioelectric potentials from the body surface.

A high-impedance input stage minimizes the effects of any imbalance or excessive amount of the electrode impedances. However, this solution presents another problem: capacitive coupling between the electrode cables and the power line. A solution is to design what is known as an active electrode, as shown in Figure B.

An active electrode typically consists of a unity gain amplifier with a

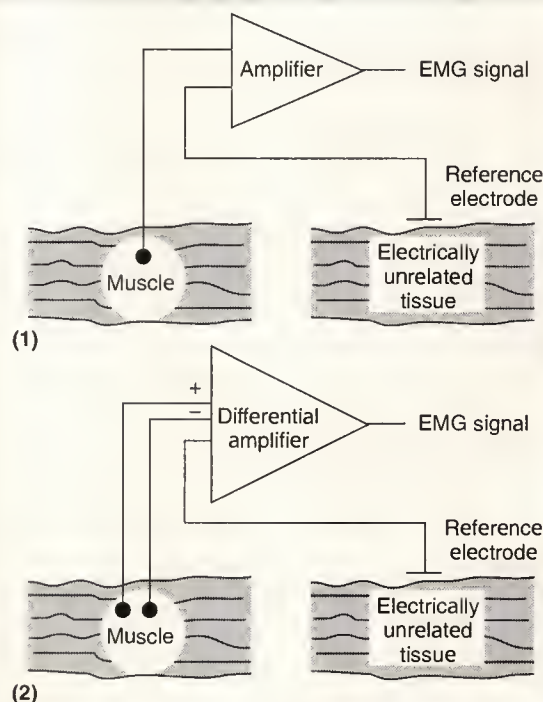


Figure A. Monopolar (1) and bipolar (2) electrode arrangements.

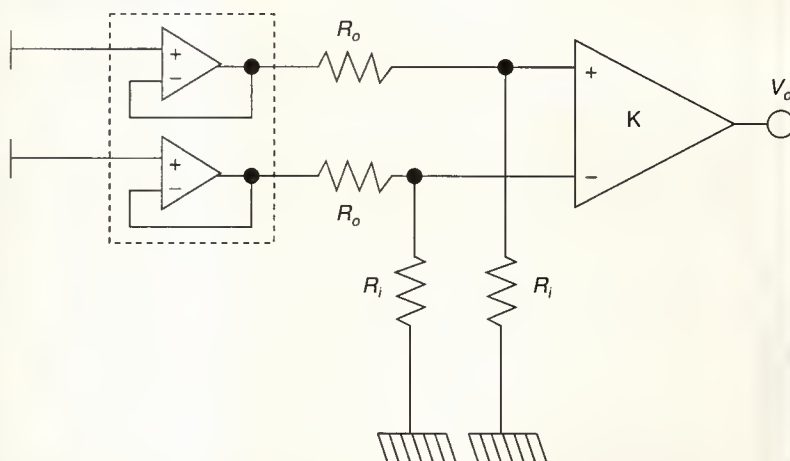


Figure B. Active detection probe principle. R_o represents the output resistance of the voltage follower. R_i represents the input resistance of the differential amplifier.

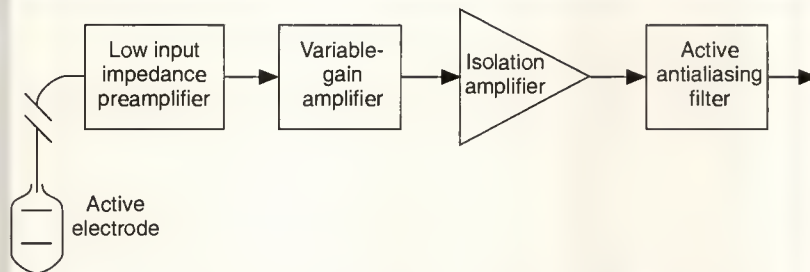


Figure C. Schematic representation of the myoelectric signal amplifier chain.

very high input impedance, mounted as close as possible to the detecting surface, thus considerably reducing the capacitive couplings between the line and the high-impedance section of the circuit. The signal is then fed to the EMG amplifier on a low-impedance path, which is much less sensitive to capacitive couplings.

Thanks to their high input impedance, active probes may be used without skin preparation, shaving, and conductive paste. Moreover, because of their high immunity to power line interferences, active electrodes can be applied in electrically noisy environments (such as hospitals) without many precautions.

The amplifier chain. Figure C shows the myoelectric signal amplifier chain. The active circuitry in the *active electrode* performs an impedance transformation. The resulting high input impedance at the detecting surfaces is converted into a low output impedance by voltage followers or noninverting amplifiers. The *low input impedance preamplifier* is generally a differential amplifier. Its low input impedance (properly driven by the active electrode) lessens the effects of capacitive couplings between electrode cables and the line. The *variable-gain amplifier* lets the user select different gains to maximize the signal-to-noise ratio during each recording.

The main purpose of the *isolation amplifier* is to isolate the stages directly connected to the patient from the rest of the circuitry and from the common ground. We currently use high-linearity optical isolators. Because of their intrinsic characteristics, optical isolators are an important source of noise in the amplification chain. For *active antialiasing filters*, fourth- or sixth-order low-pass filters should be used with a resulting roll-off of 80 to 120 decibels per decade.

Table A lists typical values of the main characteristics of myoelectric signal amplifiers.

Table A. Values for myoelectric signal amplifiers.

Characteristic	Value
Input dynamic range	0-10 mV _{pp}
Overall gain	500-10,000 V/V
CMRR*	≥ 100 dB
Input-referred noise	≤ 10 μV _{pp}
Low-frequency limiting	1-20 Hz
High-frequency limiting	250 Hz - several kHz

*Common mode rejection ratio

References

1. J. Basmajian and C.J. De Luca, *Muscles Alive*, 5th ed., Williams and Wilkins, Baltimore, 1985.

limited to 500 Hz, while the power spectrum of the signal collected by needles extends to 5 or 10 kHz.

- Since surface probes detect the action potentials generated by several tens of motor units, interpretation of SMES in the time domain is much more complicated than that of a signal collected by indwelling techniques. This difference explains why physicians showed scarce interest in SMES analysis until a few years ago.

The main advantage of SMES analysis compared with clas-

sic indwelling techniques is that surface electrodes are not invasive. Moreover, because of the relatively large detection volume of surface electrodes, SMES yields global information about muscle function.

Computer technology has had a dramatic impact on SMES analysis, because no clinical application of it would be possible without computers. In the following, we review several innovative applications of technically and clinically significant SMES analysis techniques.

continued on p. 48



Alphorn:

A Remote Procedure Call Environment for Fault-Tolerant, Heterogeneous, Distributed Systems

Alphorn (pronounced Alp-horn) is a software environment for programming distributed computer systems. Programs running on different computers, possibly of different types and running different operating systems, communicate in a client-server relationship by means of remote procedure calls. This efficient construct structures programs neatly and relieves programmers of caring about where their modules will run.

Hans-Ruedi Aschmann

Niklaus Giger

Elisabeth Hoepfli

Peter Janak

Hubert Kirrmann

**Asea Brown Boveri
Research Center**

Distributed computer systems like those used for industrial process control are large and complex. Although connecting computers in a local area network is relatively easy, programming them requires skilled personnel, especially when the project becomes large and must be extended at runtime. The tools developed by mainframe manufacturers for single-processor machines fall short in distributed systems. In particular, they do not meet the important requirements of industrial control systems: distribution, real-time response, and fault tolerance.

Development environments that support the whole software cycle, integrating programming, distribution, configuration, communication, and debugging, are appearing. They try to relieve the programmer of dealing explicitly with distribution, real-time behavior, or replication.¹ Alphorn, a programming environment developed by our

group, goes further in that direction: an application consists of a collection of modules, programmed in standard Modula-2, for instance.² These modules are written as if they were to be linked to form a single task running on a single processor. Indeed, that is the case when no distribution is required.

All communication between modules takes place through procedure calls, an approach that gives programmers a familiar view of the system. Modules interact on a client-server basis. A module exports a number of procedures, which are the services that client modules will call. The internal data structures of the modules are not remotely accessible. Programmers must obey the rules of data encapsulation.

In a distributed system, the server and client modules run on different computers, different processors of the same computer, or different tasks of the same processor. Local procedure calls be-

come remote procedure calls, but for the programmer, little is changed. Copies of a module may be loaded on different nodes to provide several instances of a service—for instance, to support several printers or to provide fault tolerance. An automatic generator makes the services of a local module remotely accessible through a stub mechanism. To this effect, the services of a module are expressed in a service interface language, which is an extension of Modula-2.

The distribution of modules on the processors and tasks is decided only at configuration time, when the work load and location of the processors are known. An interactive configurator generates the “bag” for the modules and builds the tasks. Each task includes runtime support that starts the clients, adapts the data presentation, and controls communication. The data travel over the available media: common memory, local area network, or open network. The remote procedure call (RPC) protocols are used all along, even for communication at the lowest level.

Alphorn has been ported from PDP 11/RSX to VAX/VMS, Sun/Unix, VME/VersaDOS, IBM PC, Honeywell/GCOS6, and to specialized programmable logic controllers. Although these machines use different processors, languages, and compilers, they communicate transparently with the same mechanism. Alphorn has proved useful as a structuring tool for large applications as well as a communication tool over a variety of networks.

Distributed process control

A typical distributed control system consists of computer nodes interconnected by a local area network such as Ethernet or token bus (IEEE 802.4). The nodes may be of different types and run different operating systems. They may themselves consist of multiple processors interconnected by a parallel bus and sharing a common memory. In process control, the computers are assigned dedicated tasks, as Figure 1

Alphorn has proven useful not only as a communication tool but also as a structuring tool in large projects.

RPC protocols have proven their efficiency in process control. They can be made more efficient if the medium provides broadcast. However, the Open System Interconnection (OSI) protocols of the International Standard Organization (ISO) do not support broadcast communications. To support efficient replication and configuration, Alphorn uses a causal broadcast protocol, implemented on top of available protocols like TCP/IP. In fault-tolerant applications, this protocol supports the actualization of replicated services transparently.

Several other projects at least partly explore the same approach. The ISIS system, which set the bases of RPC communication, provides a form of replication.³ Apollo's Network Computing System and Sun's Network File System are commercial products that use RPC communication, but they do not consider fault tolerance.^{4,5} The Advanced Networked Systems Architecture (ANSA) project and the ESPRIT Delta-4 project define a general framework for distributed systems but have a weak emphasis on fault tolerance.^{6,7}

These projects influence the current standardization work of ECMA⁸ (European Computer Manufacturers Association) and especially the work on Open Distributed Processing (ODP) supported by ISO/IEC SC21 and the CCITT. These standards define an open architecture with many options and variants. By contrast, Alphorn ensures communication within a dedicated system, allowing us to keep it simple.

(on the next page) suggests.

The communication takes different paths, depending on the location of the partners:

- *Nodes* communicate through the network by sending and receiving messages. They do not share a common address space. A name server may provide a systemwide directory service.
- *Processors* within a multiprocessor communicate through a common memory, which may be located globally on the parallel bus or may consist of dual-ported local memories. The processors may take advantage of the message-passing facility offered by some parallel buses, such as Multibus II. They may form a pool or be of different types.
- *Tasks* executed by the same processor share the same physical memory but are prevented from accessing each other's region by the memory management unit. They communicate through shared-memory regions using services of the kernel. (Tasks are called processes in VAX/VMS or Unix terms.)
- *Threads* are parallel activities within the same task. Threads (called processes in Modula-2 and light-weight processes in Unix) are based on coroutines. They share the address space of their task and communicate with

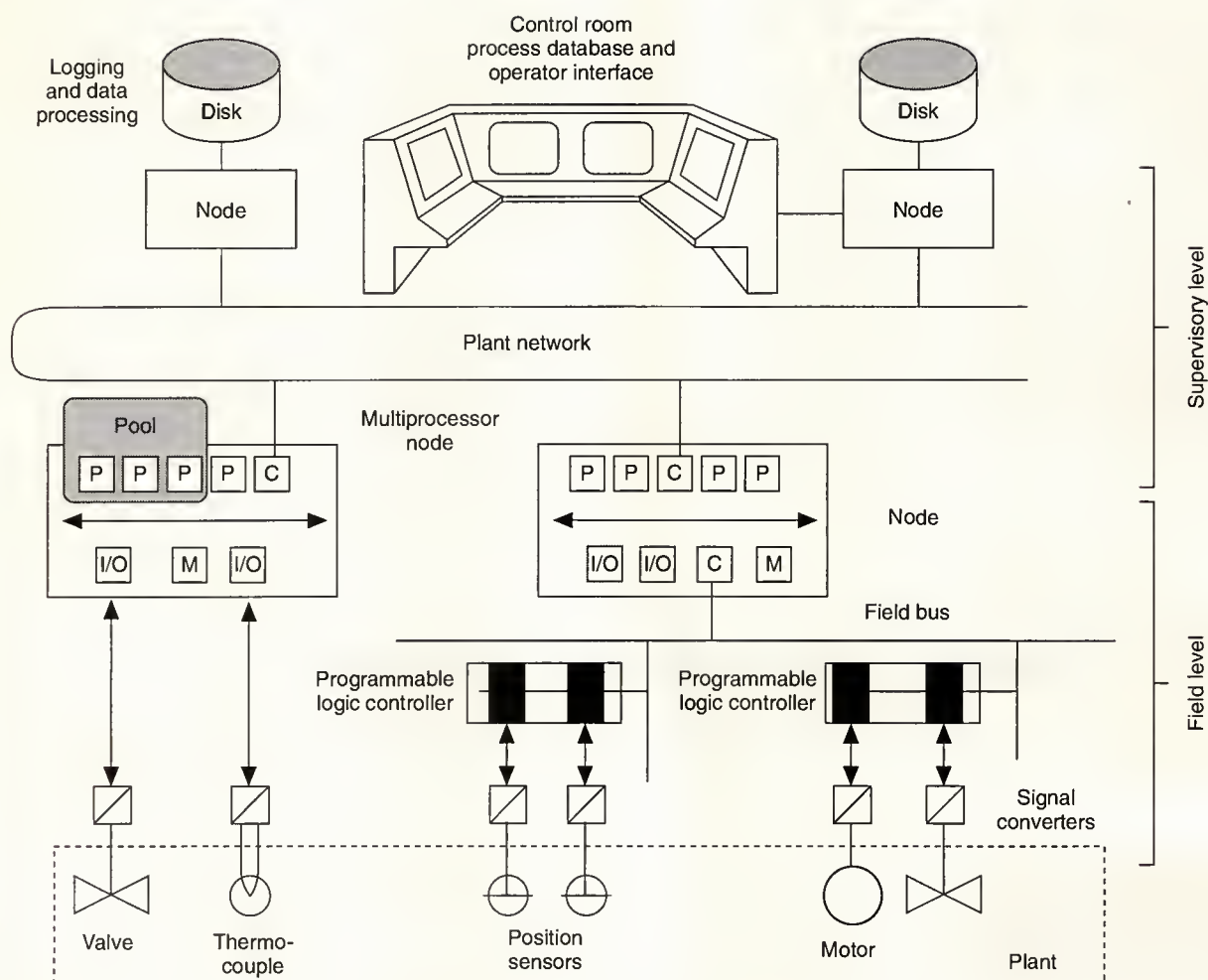


Figure 1. A distributed process control system.

each other through common variables. Threads of the same task cannot interrupt (preempt) each other, but a thread runs until it reaches a synchronization point. The task holds a small scheduler that determines the next thread to run. Because of its synchronous nature, a thread switch is faster than a task switch. (Threads are the smallest unit of parallelism we consider here.)

The communication between these entities is illustrated in Figure 2. Since communication paths vary widely, we wish to offer the programmer a unified communication interface independent of the communication path.

Traditionally, communication between remote entities has

been based on messages. Logical channels, which implement point-to-point or multipoint links over the same physical medium, carry the messages. Channels are called mailboxes, pipes, ports, and so on, with subtle variations in meaning. The programmer first opens a channel and then sends data explicitly by using something like a Send Message operation. The partner receives the message by a Receive Message operation. These operations can be inserted at any place in a program, allowing great freedom but also great responsibility to the programmer. Message passing lends itself to spaghetti-code programming, which is hard to debug and maintain, especially when several entities write and read to the same channel.

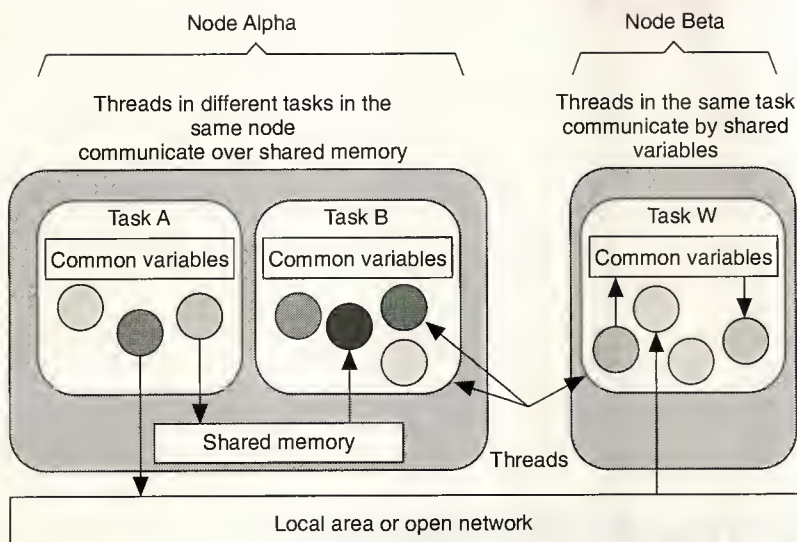


Figure 2. Parallel entities.

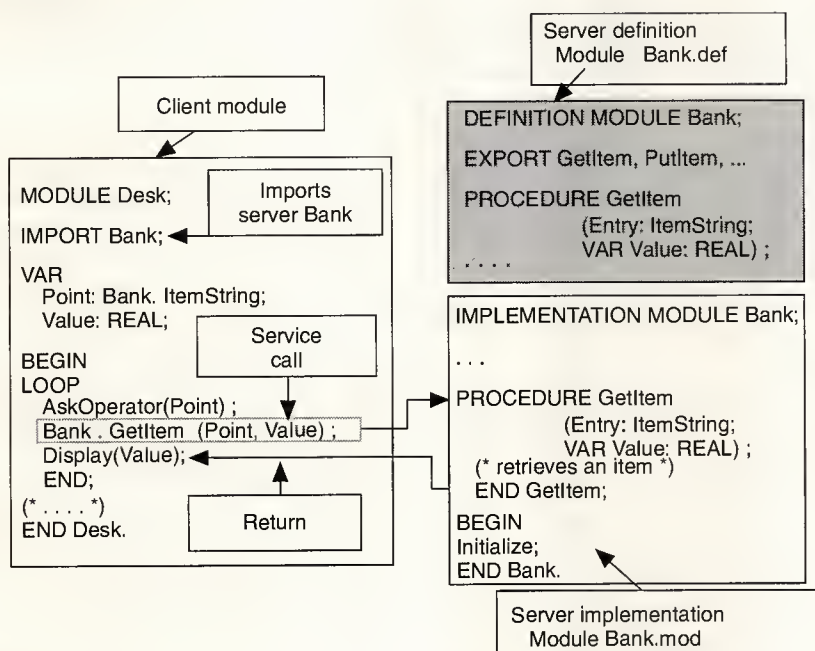


Figure 3. Calling a service from a server module.

While the communication system cannot avoid some kind of message passing, the application programs should be better structured. Alphorn relies on the Remote Procedure Call (RPC) as an alternative to message passing. The difference between messages and RPCs is comparable to that between Fortran and structured languages without Goto statements. Programming with RPC considerably influences programming style. When RPCs are properly used, there is no need for other communication constructs.

Programming style

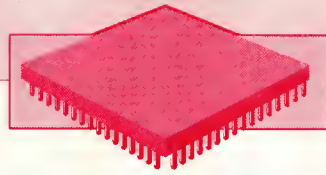
Programming in Alphorn is based on the concept of objects, which is supported by high-level languages. An object consists of data structures and a well-defined set of procedures giving access to the data. An example of such an object is the Oracle database; the contents of the database can be accessed only through procedures such as GetItem, InsertItem, RemoveItem, and SearchItem. These procedures form the object interface, which is all a user must know to access the database. The database can be implemented by arrays, lists, records, and so on, which the user need not know about.

Modula-2 expresses the object interface in the definition module, which lists all exported variables, data types, and procedures. The implementation of the object is contained in the implementation module, which is a private affair of the object. The database user ignores how and where the items are stored.

A module that exports procedures becomes a server when called by a client program. An example of a service call appears in Figure 3. The server "Bank" is defined by a server module. Its server interface is defined in the definition file "Bank.def." The procedures exported by the definition module (for example, GetItem) are the services of that server module. The services are programmed in the implementation file "Bank.mod."

The calling conventions of the client

continued on p. 60



A RISC Processor for Embedded Applications Within an ASIC

RISC processors are generally not suitable for use as ASIC cores due to their die size, power dissipation, and pin counts. The VL86C010 processor, however, is well suited to RISC applications because of its small die, low power dissipation, and low pin count.

Charles E. Roberts

VLSI Technology, Inc.

Speed is the ultimate challenge in computing—it always has been and it probably always will be. Nowhere is this more true than in embedded controller applications in which application-specific ICs (ASICs) are often considered for the design solution. However, while many speed-sensitive embedded controller applications require high performance, they also experience the penalties such performance levels create—penalties that significantly affect ASIC designs.

Designing a control system to embed in some larger product or system usually frustrates designers because of the engineering demons of size, weight, power, cost, reliability, and security. Designers need a fast, inexpensive, low-power controller that can handle the most functions with the least hardware. Low power is important to reduce the size, weight, and cost of the power supply and reduce the requirement for special heat control devices (heat sinks, fans). Unfortu-

nately, speed, power, and cost are usually opposing parameters, especially in products for the reduced instruction-set computing (RISC) processor market.

To achieve RISC performance, designers expect to pay heavily in terms of power and cost. System integration using one or more ASICs is the logical route to achieving the goals of size, weight, power, and cost reduction, and of increasing system reliability and security. However, most RISC processors currently available have die sizes, power requirements, and pin counts that make them economically infeasible for use in ASIC designs. In many cases these RISC processor die sizes reach the upper limit of present manufacturing capability for practical ICs, and their prices reflect this fact.

Our high-performance RISC processor family offers the power, cost, die sizes, and pin counts that are well suited to ASIC designs. Designed using our IC design tools, these parts naturally fit

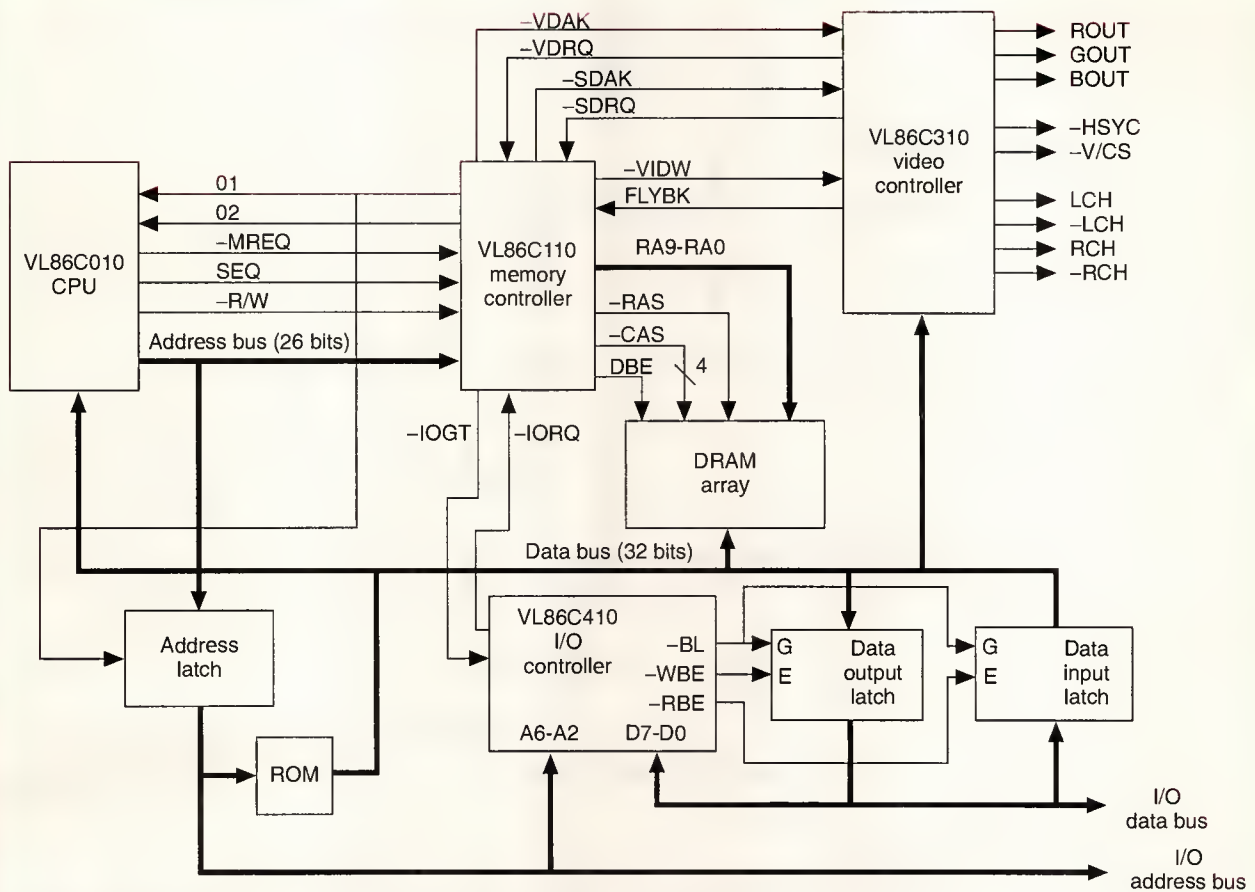
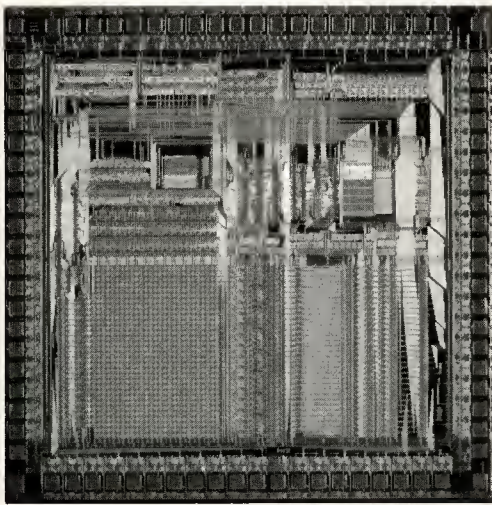


Figure 1. VL86C010RISC system block diagram.



VLSI's VL86C010 32-bit, fixed-point RISC processor comes in an 84-pin PLCC package and dissipates 100 mW.

as cores in other ASIC designs that use the same design tools. The architecture, instruction set, and features show that this family is powerful in its own right, and especially so for embedded control applications.

A significant difference between this RISC processor family and others is that it was designed for maximum system performance at minimum cost rather than maximum CPU performance at any cost. Designers took a fresh approach to the system design and partitioning to provide high performance and very high speed interrupt response while keeping the system busing to a minimum (see Figure 1). With these goals in mind, they partitioned the components of the system (CPU, memory controller, I/O controller, and video controller) to take advantage of less-expensive plastic IC packages and, more importantly, less-expensive dynamic RAM devices (DRAMs) by using their page mode of operation.¹

This partitioning allowed the CPU to be placed into an 84-pin plastic leaded chip carrier (PLCC) and the other members of the family to be placed into 68-pin PLCCs. It also reduces the number of bus lines on a custom ASIC device, one key to

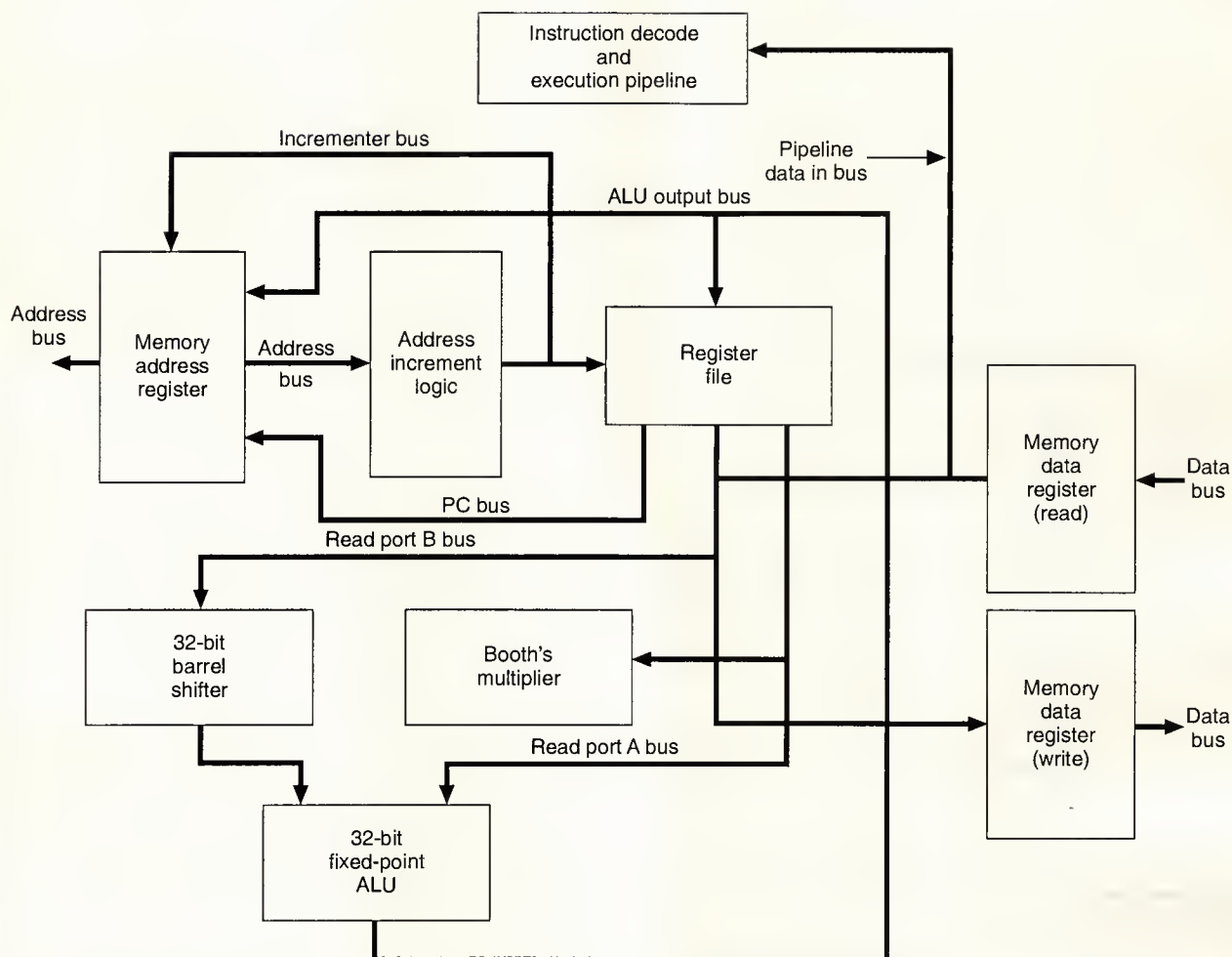


Figure 2. VL86C010 RISC processor block diagram.

the devices' suitability as cores. The reduced number of bus lines and wires to route has a corresponding impact on die area and, so, on cost. On some large, complex chips, it is not uncommon for designers to use close to 50 percent of the area for routing.

The CPU

The RISC CPU (VL86C010) is a 32-bit, fixed-point processor with a barrel shifter and a Booth's multiplier, as seen in Figure 2. It contains twenty-seven 32-bit registers (organized as four partially overlapped sets of 16 registers), a three-level instruction pipeline, a 64-Mbyte linear address space, and full virtual memory support. Even though this processor produces impressive performance statistics, it typically draws just 0.1 watts of power at 10 MHz. One particularly interesting

feature of the instruction set is that every instruction can be executed conditionally in the same manner as branch instructions.

The processor has 16 possible conditions (such as zero, positive, overflow) coded in the opcode in a 4-bit field. Because the hardware was already there to decode this field for the conditional execution of branch instructions (like virtually all processors), designers found it an easy matter to make all instructions conditionally executable. As a result programmers can execute two different segments of code depending upon a condition without making a conditional branch over one segment of code. Programmers simply include the condition directly in the instructions of that segment. For short forward references (which are very common), this approach yields more compact and faster code.

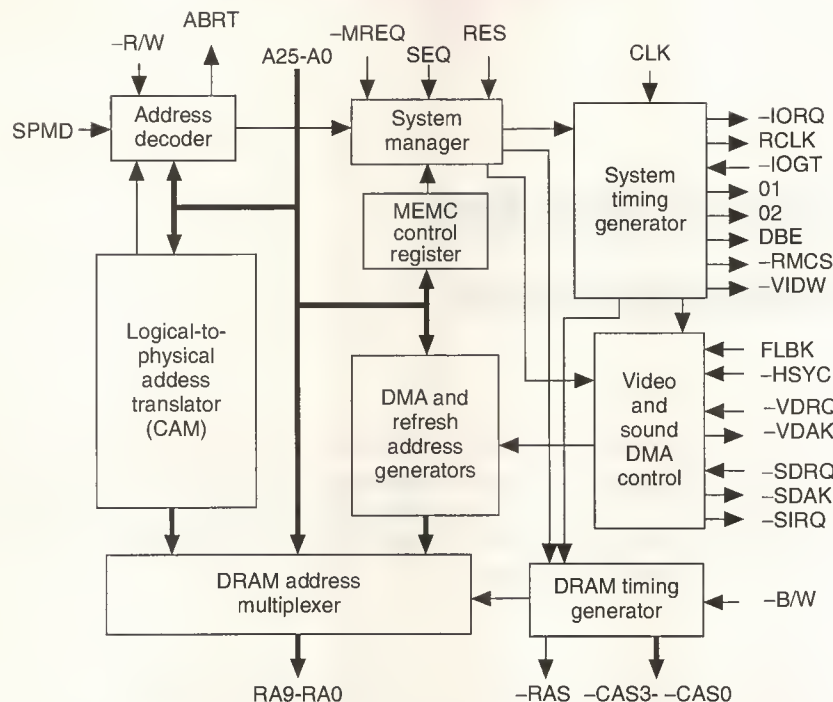


Figure 3. VL86C110 MEMC block diagram.

This and other features (discussed later) allow this processor to outperform other processors clocked at higher rates. The higher clock rates produce higher power and faster memory requirements that are undesirable for embedded control applications. We offer our RISC processor at competitive clock speeds but find it frequently doesn't need to be run that fast because of its high throughput.

No matter how fast a processor can execute internal instructions, it loses value quickly if it doesn't have an equally fast method of communicating with the outside world. For embedded applications, this concern is often most important in handling interrupts. The VL86C010 contains a classical interrupt request input pin that behaves as on most other processors. In addition, the processor features a *fast* interrupt request input pin.

When this pin is pulled active (low), the processor finishes the current instruction, switches the mode to fast interrupt, and switches register sets (picking up seven new registers dedicated to this mode). It immediately executes the fast interrupt subroutine at a dedicated memory location so that a branch instruction doesn't need to be executed. In this mode the processor can respond to the FIRQ instruction in 2.5 clock cycles. The worst-case maximum response of 22.5 cycles occurs only when block register move instructions are executing and the

size of the block is set to the maximum of 16 registers. If faster worst-case interrupt response is required, programmers simply don't make 16 register block moves in the code. As it turns out, the worst-case response is not a significant penalty as one rarely needs to move all the registers at once.

Memory controller

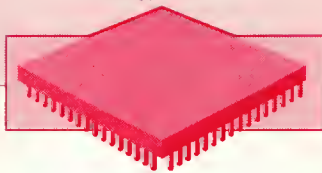
The VL86C110 RISC memory controller (MEMC) handles all the timing functions for the DRAMs and uses the fast page mode to maximize bandwidth (see Figure 3).

The MEMC contains DMA address generators for video, cursor, and sound data buffers and provides full support for logical-to-physical address translation for virtual memory. Of particular interest is the fact that the 32-bit data bus does not connect to the MEMC. This feature affects the core size of the MEMC as well as the die size of an ASIC that would use the MEMC because of the reduced busing discussed earlier. While speed is all important in RISC applications, die size is all important in ASICs because of the cost.

Video controller

The VLSI VL86C310 video controller (VIDC) contains all the circuitry to implement color video and stereo sound in-

continued on p. 68



Performance and the i860 Microprocessor

The internal design of the i860 CPU exploits pipelining and parallelism more than previous microprocessors. It uses RISC concepts and memory-performance optimizations in several novel ways. Understanding these features will help programmers optimize code and system designers maximize throughput.

Mark Atkins

Intel Corp.

In 1989 Intel introduced the i860 XR (Figure 1) processor as the fastest microprocessor possible in a million-transistor chip.^{1,2} It performs at 40 MHz without external cache memory at 26.7 Specmarks. Contributing to this speed are instructions tuned for parallelism, internal caches with a 960-Mbyte/s internal bandwidth, and an 8-byte external bus operating at 160 Mbytes/s. In 1991, Intel introduced the i860 XP microprocessor, a larger, faster version with added features. Most of the topics I discuss here apply to both chips.

Architecture versus implementation

People generally define a computer's architecture as the programmer-visible instruction set, registers, and memory arrangement. Most architectures have a variety of different implementations (actual hardware boxes or chips conforming to the architecture) at various cost levels.

However, architecture and implementation are closely intertwined. We can't run benchmarks on architectures, but an obsolete architecture can slow implementations. For example, the complexity of the IBM 370 renders a given amount of hardware slower than the same hardware devoted to a RISC (reduced instruction-set computer) machine.³ Computer buying and design decisions usually consider implementation more than architecture. In addition, possible hardware

implementations influence initial specification of the architecture. The feasibility of a million-transistor chip allowed the i860 designers to include features like virtual memory, floating point, and 3D-graphics support.

The architectures of most RISCs are very similar. All attempt to give the best possible performance per dollar, using:

- 32-bit instructions, data, and addresses;
- large register files, to avoid (slow) memory accesses;
- pipelining;
- caches for fast instruction and data access;
- register-to-register operations, to avoid memory accesses;
- "reduced" number of instructions, to avoid long design periods, debugging efforts, and silicon space for microcode; and
- delayed branches, which execute the instruction after the branch to accomplish useful work while fetching the branch target.

While the i860 CPU implements all the RISC rules, it adds several novel items to enhance performance. Even when software does not exploit the extra parallelism features, the i860 still achieves efficient RISC performance. As compilers grow more sophisticated, they may produce faster code through simultaneous parallel execution of integer and floating-point (or graphics) instructions.

The i860 CPU implementation

The most notable difference between the i860 XR processor and other CPUs is its single-chip implementation. This one piece of silicon contains:

- an integer unit, including thirty-one 4-byte registers;
- 32 floating-point registers, an adder unit, and a multiplier unit;
- a graphics unit;
- a 4-Kbyte instruction cache;
- an 8-Kbyte data cache; and
- a memory management unit with 64-entry translation look-aside buffer.

This approximately \$500 chip makes systems less expensive in both chip cost and system engineering effort than a separate integer chip, floating-point chip, cache chips, and cache controller chip found in competing RISC machines. Most chip suppliers now are working on single-chip solutions to be introduced in 1991.

Beyond cost considerations, a single chip offers advantages from wide, fast buses, which are feasible only when they do not cross chip boundaries. The on-chip cache bandwidth, which is 960 Mbytes/s at 40 MHz (Figure 2), includes:

- a 320-Mbyte/s instruction bandwidth (two executed each clock cycle) and
- a 640-Mbyte/s data bandwidth (16 bytes fetched or stored each cycle).

While the 12 Kbytes of on-chip cache is less than would be possible with separate external cache chips, it contributes to performance significantly with a hit rate typically exceeding 90 percent.

Multiport register files. The register file also is crucial to on-chip bandwidth. For example, the 32-bit ports on some processors reduce double-precision, 64-bit floating-point performance by requiring two accesses to secure one operand. In the i860 CPU, the floating-point registers' five 64-bit ports can fetch three double-precision numbers in each

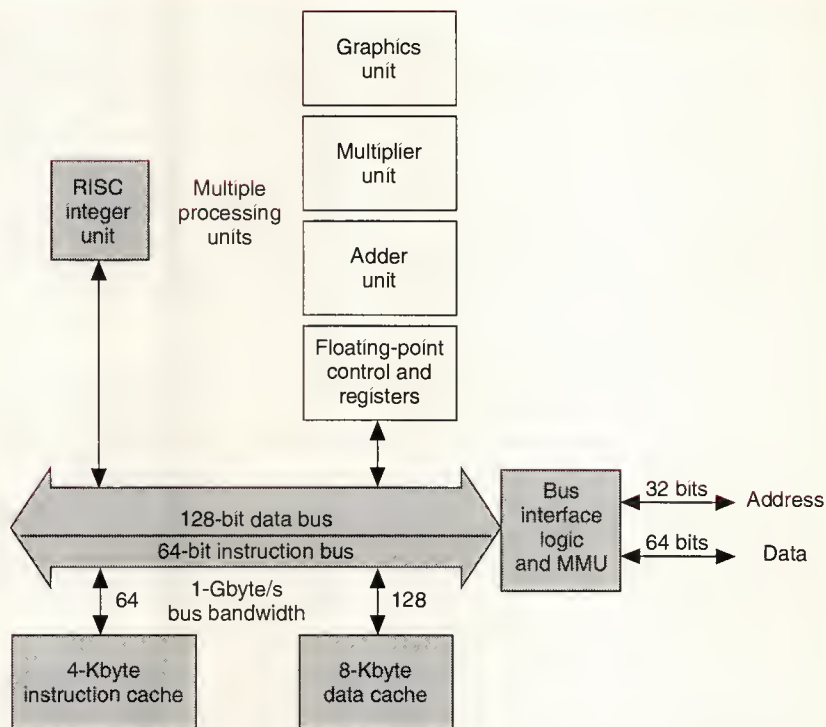


Figure 1. The i860 XR processor block diagram.

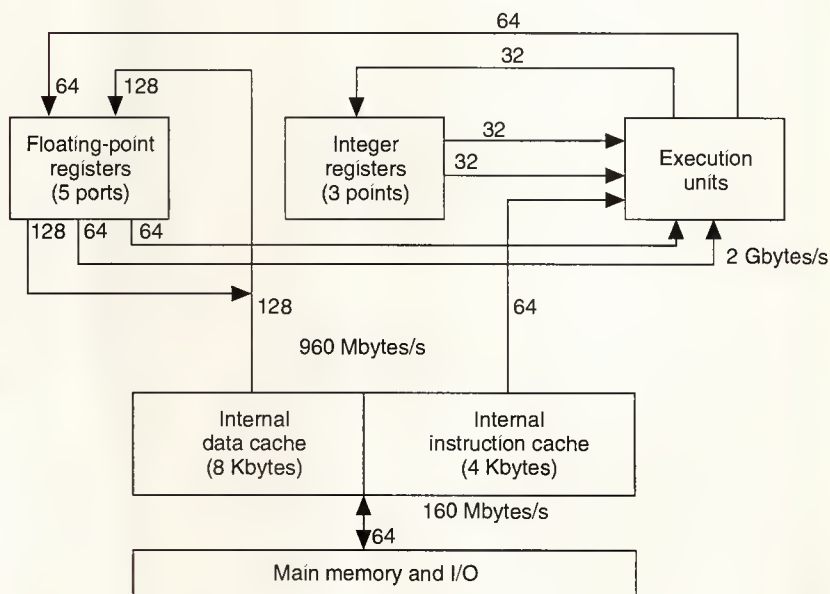


Figure 2. Internal and external bandwidths at 40 MHz.

clock cycle—two numbers for an add (or multiply) operation simultaneously with one for a store-to-memory step. Simultaneously, the register file can write the result of a previous add (or multiply) and the result of a memory load. The integer registers contain two read ports and one write port, allowing the simultaneous three-operand accesses implied in instructions such as ADDU R1,R2,R3. The register bandwidth totals 2 Gbytes/s.

Pipelining. The instruction processing pipeline of the i860 processor, shown in Figure 3, includes four stages assigned the acronym FDEW:

- fetch the instruction from the cache or memory;
- decode it and access the register file(s);
- execute the arithmetic, shift, or logical operation; and
- write the result into the register file.

Those four segments of processing usually require similar amounts of time due to hardware considerations, and thus most RISC processors use a four-stage pipe. Figure 3 is a pipeline resource diagram, plotting time against hardware and instructions. Such diagrams assist designers in avoiding resource conflicts. By drawing sequences of instructions and their hardware usage, designers can detect conflicts such as overlapping register-port usages. As long as the same hardware (such as the ALU, abbreviated E in the diagrams) is not used by two instructions simultaneously, the pipeline works.

The pipeline for both conditional and unconditional branches works in four stages abbreviated FDIW, similar to FDEW:

- fetch the (branch) instruction;
- decode and generate the branch target address;
- access instruction cache for target and decode the instruction in the branch delay slot (if conditional branch, check condition code bit to determine if taken); and
- write the return address to the register file (if this is a subroutine call) and execute the instruction in the branch delay slot.

If a conditional branch (BC.T, BNC.T) is not taken, the pipeline takes an additional clock cycle to restart the fetch of the sequential stream. Of course, if the sequential instruction is a cache miss, or the target is a miss for a taken branch, the CPU initiates an external bus fetch and suspends execution.

A dedicated branch adder unit calculates the target address using the current program counter and an offset amount embedded in the current instruction. The 26-bit offset denotes 4-byte instructions, giving a 256-Mbyte direct target range. To allow a full 4-Gbyte branch range, the CPU also implements indirect branches whose target comes from a 32-bit register.

The on-chip instruction cache shortens the branch pipe-

Code fragment:
 addu src1,src2,dest
 br target
 subu src1,src2,dest
 (more instructions here from sequential stream)

target:
 subs src1,src2,dest
 ...

(a)

ADDU	F	D	E	W			
BR (branch to target)		F	D	I	W		
SUBU (delay slot)			F	D	E	W	
SUBS (target)				F	D	E	W
Clock cycle	0	1	2	3	4	5	6

(b)

Figure 3. Code fragment (a) and instruction pipeline (b).

line to four stages, when it might otherwise be five, with two clock cycles required in the I stage of FDIW. Were the cache external, the processor would have to drive the address on the external bus, propagating through the external cache and transferring the instruction via the external bus.

Floating-point pipelines. While RISC implementations universally follow the FDEW scheme for integer operations, they vary much more in the floating-point arena. Here the i860 CPU excels because it implements a low-latency and high-throughput three-stage execution pipe. Low latency is essential to performance when executing serial algorithms, in which each calculation uses results from the previous calculation. The floating-point adder and multiplier each have three stages, which amount to 75-ns latency at 40 MHz. Actually, the multiplier uses two stages at two clocks each for double-precision operands.⁴ The graphics instructions also use the floating-point registers for additions and comparisons on integer values, but are faster at one clock cycle each.

Including fetch (F), decode (D), and write-back (W) with the three execute (E) stages yields six stages overall. Two versions of floating-point instructions exist, as illustrated in Figure 4:

- scalar, in which the E stages of successive floating-point operations do not overlap, and
- pipelined, in which the E stages do overlap.

Note from the figure that the integer and floating-point operations can overlap; only consecutive scalar floating-point instructions cannot overlap the E stages. The separate floating-point and integer register files allow writes to be


```

addu src1,src2,dest1    // Integer
fadd fsrc1,fsrc2,fdest2 // Floating-point
fsub fsrc1,fsrc2,fdest3 // "
fmul fsrc1,fsrc2,fdest4 // "
subu src1,src2,dest5    // Integer

```

(a)

ADDU	F	D	E	W									
FADD		F	D	E1	E2	E3	W						
FSUB			F	D	—	—	E1	E2	E3	W			
FMUL				F	—	—	D	—	—	E1	E2	E3	W
SUBU							F	—	—	D	E	W	
Clock cycle	0	1	2	3	4	5	6	7	8	9	10	11	12

(b)

ADDU	F	D	E	W				
PFADD		F	D	E1	W _E	E3		
PFSUB			F	D	E1	W _E	E3	
PFMUL				F	D	E1	W _E	E3
SUBU					F	D	E	W
Clock cycle	0	1	2	3	4	5	6	7

(c)

Figure 4. Floating-point instruction pipeline: Code fragment containing both integer and floating-point operations (a), scalar instructions (b), and pipelined instructions (c). A dash indicates an idle clock cycle.

processed out-of-order as in the case of the FMUL (floating-point multiply) and integer SUBU (subtract) in the figure. The pipelined floating-point instructions write the result from a previous instruction simultaneously with the second E stage, creating an FDEW_E sequence.

Explicit pipelines. The pipelined mode is similar to microcode on array processors in which the instruction specifies two source operands but does not specify the corresponding destination register. Instead, the destination field of the pipelined floating-point operation receives the result from the third previous floating-point instruction. That is, floating-point instruction *I* updates its RDEST with data generated by floating-point instruction *I* - 3. This explicit pipelining makes programming a bit tricky because the programmer or compiler must keep track of the contents of the floating-point pipelines. However, it also yields a 3-to-1 speedup for floating-point-intensive software. At 40 MHz, scalar code achieves 13 Mflops (million floating-point operations per second), but pipelined code yields 40 Mflops. Software libraries for i860 graphics and vector math exploit pipelined floating point, and compilers have begun to generate it.

Additionally, the explicit pipelining keeps more floating-point registers available to the program. If FADD F5,F6,F7

updates F7 with the sum F5 + F6, then F7 cannot be used in the next three instructions without causing a hardware interlock (delay) for those instructions to wait for the correct F7 value to emerge from the execution pipeline.

Finally, explicit pipelining simplified the hardware implementation, removing the need for controls to check for a just-mentioned case (instructions attempting to use a floating-point register whose contents are obsolete). The avoidance of that checking logic is also the reason that scalar floating-point instructions wait for their predecessors to finish the E stages. The name "scalar" denotes the type of code in which the destination register of an instruction becomes a source in the next instruction, in which case the interlock is necessary anyway.

Even without architectural changes, scalar throughput can improve to one clock cycle in future implementations that provide interlock checking. Then both scalar and pipelined code can achieve 1 Mflops per megahertz.

Some computer architects criticize the exposed floating-point pipelines and the embedding of their length into software, because future technologies might allow shorter pipes. However, any software contains implicit assumptions of pipeline length, as the instruction scheduling for avoidance of interlocks requires latency knowledge. Like the Mips R2000/R3000 architecture with the memory-load pipeline length exposed to software, the floating-point exposure allowed simpler hardware and software optimization. And like Mips-2 with the load-delay slot removed, future considerations may require redesigning the floating-point pipelines. However, a compatible three-stage floating-point mode will always be supported in the hardware to run old "dusty deck" code, that is, old programs that current users cannot, or will not, rewrite.

Duality and six-stage pipelines. To avoid unnecessary instruction sequencing delays, the chip incorporates two features. I use "unnecessary" to refer to the time the floating-point unit idles while integer instructions execute (or vice versa), and the time the multiplier idles during adder operation.

One feature, dual-operation floating-point instructions, allows both the floating-point adder and multiplier to work in multiply-accumulate fashion. This kind of parallelism is common in digital signal processor chips and complex numbers. In dual operations, the three-stage adder and multiplier pipe-

continued on p. 72

Information for Authors

June 1991

Who we are

IEEE Micro, a bimonthly publication of the IEEE Computer Society, reaches an international audience of microcomputer and microprocessor designers, system integrators, and users. Readers seek to increase their technical knowledge of computers and peripherals; systems, components, and sub-assemblies; communications, instrumentation, and control equipment; and software.

What we publish

IEEE Micro publishes original works about 5,500 words long (about 20 double-spaced typed pages that include explanatory figures, tables, and programs). These works discuss the design, performance, or application of microcomputer and microprocessor systems. Readers welcome tutorial material, review papers, and discussions of standards. Topic areas include:

- systems
- fault tolerance
- languages
- application software
- algorithms
- hardware and software design and implementation
- architecture
- data acquisition
- operating systems
- artificial intelligence
- communications

Submitting your manuscript

Submit six copies of your manuscript and a 50-word abstract with keywords along with your mailing address, phone and fax numbers, and electronic mail address directly to:

Prof. Dante Del Corso
Editor-in-Chief, *IEEE Micro*
Dipartimento di Elettronica
Politecnico di Torino
C.so Duca degli Abruzzi, 24
10129 Torino, Italy
Telephone: + 39 11 5644044; fax: + 39 11 5644099

Compmail: d.delcorso; Bitnet: delcorso@itopoli;
Internet: delcorso@polito.it

or

Ashis Khan

Associate Editor-in-Chief, *IEEE Micro*

Mips Computer Systems, Inc.

950 DeGuigne Drive

Sunnyvale, CA 94086

(408) 524-7171

Internet: ashis@mips.com

All manuscripts pass through a peer-review process consistent with other professional-level technical publications. This process may take up to four months, and referees may require revisions to parts of your work. If a manuscript exceeds the specified length, it will be shortened.

Successful contributions avoid the style of transactions and academic journals. They sufficiently introduce the material, place it in context with similar works, describe the practical or potential applications of the material presented, and discuss both pros and cons of the approach. At least 20 percent of the article is tutorial in nature. Brief literature surveys do not satisfy this requirement.

After accepting your manuscript for publication, the editor-in-chief will ask you to supply three copies of any revised draft, plus drawings, photographs, equations, and programs; an electronic version; and biographies and photos of all authors. In addition, you must sign a release transferring copyright to the IEEE (excepting certain key rights retained by the author).

Submit the hard copies, including illustrations and references or bibliographies, printed on one side only of 8 1/2 x 11-inch paper and double spaced with at least 1 1/2-inch margins. Send an electronic copy on floppy disk or via Compmail or Internet. All electronic files should retain any text-formatting codes you use and identify the formatter used. Refer to the Computer Society's Electronic Submittal Guide

(attached) for further details. Disks must be Macintosh-compatible or 5.25-inch, IBM PC-compatible, and running DOS Version 2.10 or newer. For further guidance, contact:

Marie English
Managing Editor, *IEEE Micro*
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264
Telephone: (714) 821-8380; fax: (714) 821-4010
Comppail: m.e.english

Professional editors on the *IEEE Micro* staff thoroughly edit accepted manuscripts. This collaborative process between author and editor results in a concise, well-worded article.

Writing tips

Readers welcome clear, accurate articles presented in logical sequence. Let readers know in the first paragraph why your subject is important; give them a reason to continue reading. Augment your discussion with examples, tables, diagrams, charts, and photographs to help readers grasp your point. Remember, all readers won't be familiar with your specialty; you will have to explain unusual terms or intricate processes.

Readers move swiftly through articles written in the active voice and containing short words, short sentences, and concrete examples. (An active voice example: "This scheme contains two main buses" NOT "Two main buses are contained in this scheme.") Avoid jargon, explain acronyms, and simplify your language. For example, use "to" NOT "for the purpose of" and use "can" NOT "has the capability to." In other words, write the way you talk.

As you can see, magazine style differs from journal and report styles.

References and bibliographies

References substantiate points made in the text or cite previous or important works. Do not overdo it, however; most articles need less than 10 citations. They appear in numerical order in the article and in a separate section at the end of the article. Citations in the text appear as Arabic superscripts, for example, Smith.¹

Cited sources should be available to the reader; don't include unpublished works. Any abbreviations should follow *IEEE Micro* usage; see a recent issue for examples. When in doubt, spell it out.

You should attempt to provide full bibliographic data as a courtesy to your readers. A complete citation includes author(s); title of article or chapter; title of journal, book,

proceedings, or dissertation; volume; number; publisher's name, city, and state for books and dissertations; complete address for private technical reports; year published; and inclusive page numbers.

Illustrations

Submit photocopies of illustrations, rather than originals, for the initial manuscript review. Cite permission for any previously published images or figures so that *IEEE Micro* can properly credit the source. (See Figure 1.)

All illustrations and drawings should be clear and submitted in hard copy (on separate sheets) and on Macintosh-compatible electronic disks where possible. Photographic prints should have good contrast and gradation and should be at least 3 × 5 inches in size. Number, caption, and cite in text all illustrations and tables. Check to see that all artwork is accurate and unambiguous, and uses the same terms as the text.

IEEE Micro reproduces your original halftones, machine-made graphs, computer printouts, and electronically produced artwork. Artists will redraw all other art to meet house standards.

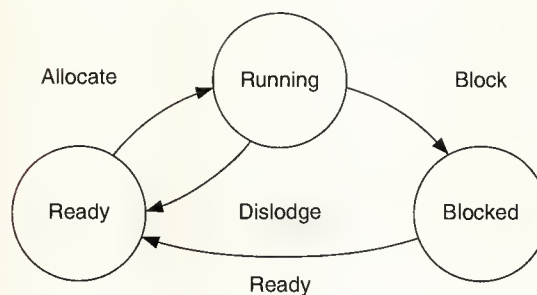
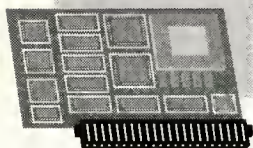


Figure 1. Task states and transitions. (Copyright 1995 William Jones. Reprinted by permission.)

Biographical sketch and photograph

Submit a photograph and biographical sketch of each author. Good-quality, black-and-white glossy photographs, preferably 3 × 5 inches in size reproduce best. Limit biographical sketches to 75 words and include, in the following order: current positions and technical interests, prior professional experience and other important activities, education, professional affiliations, and current address. See a recent issue of *IEEE Micro* for examples.

On the Edge



IEEE Standard P1754: An Open Microprocessor Architecture

[The second part of our series on tools addresses a standard ready for final balloting. Rudolf Usselmann discusses the implications of this standard and stresses its advantages. Please note the balloting date in the last paragraph.]

I invite readers to send me input regarding a tool or method that helps solve problems for future columns in the series.—C.W.]

Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 6-0669

warren@ssdunx.mdc.com

Rudolf Usselmann
Sparc International

Many of today's existing microprocessors are vendor proprietary architectures and not subject to industry consensus. In September 1990, Sparc International, Inc. initiated a P1754 (IEEE standards) working group. Its goal was to define an open microprocessor architecture that is not bound to a specific technology and is available to a wide variety of manufacturers and users.

P1754 is the first open microprocessor architecture to be accepted as a standard by the IEEE. Some people expressed an understandable skepticism about the outcome of the project due to the number of players involved. Many did not think representatives from so many major companies would be able to agree on one architecture that would be feasible to implement. After many meetings, five P1754 drafts, megabytes of e-mail, and endless phone conversations, P1754 is now in its final stage in the standard development process.

The outcome of the P1754 project is a

nonproprietary microprocessor architecture based on the existing Scalable Processor Architecture (Sparc) developed by Sun Microsystems. The standard includes a definition of the instruction set architecture (ISA), register model, data types, instruction opcodes (including floating-point instructions), trap model, coprocessor interface, and other architectural aspects.

P1754 is an architecture, not an implementation. It is very important to highlight this statement. The standard does not limit you to follow a specific method when designing a device, it rather encourages you to be creative, leaving "back doors" open so you can add your own extensions to the architecture without violating the standards definitions. Many of you will ask, What is a standard good for if it allows you to add your own extensions? Let me answer this with an example.

Say that company XYZ is implementing a P1754 CPU and the company's designers want to incorporate a graphics accelerator on the device. Now, they can use the implementation-dependent extension feature to include additional instructions and registers on their device and still be compliant with the standard. To make use of the extra functionality, the application executes a call to the operating system, which enables the extensions, executes the requested function, and then switches back to a compliant mode. In other words, system-dependent libraries, which are runtime linked to applications, can exercise the extensions totally invisible to user software.

To further emphasize the flexibility and thoroughness of the architecture, let me point out the register window definition. An implementation may have from two to 32 sets of windows with each set containing 24 registers (of which eight are shared with the next window, thus only 16

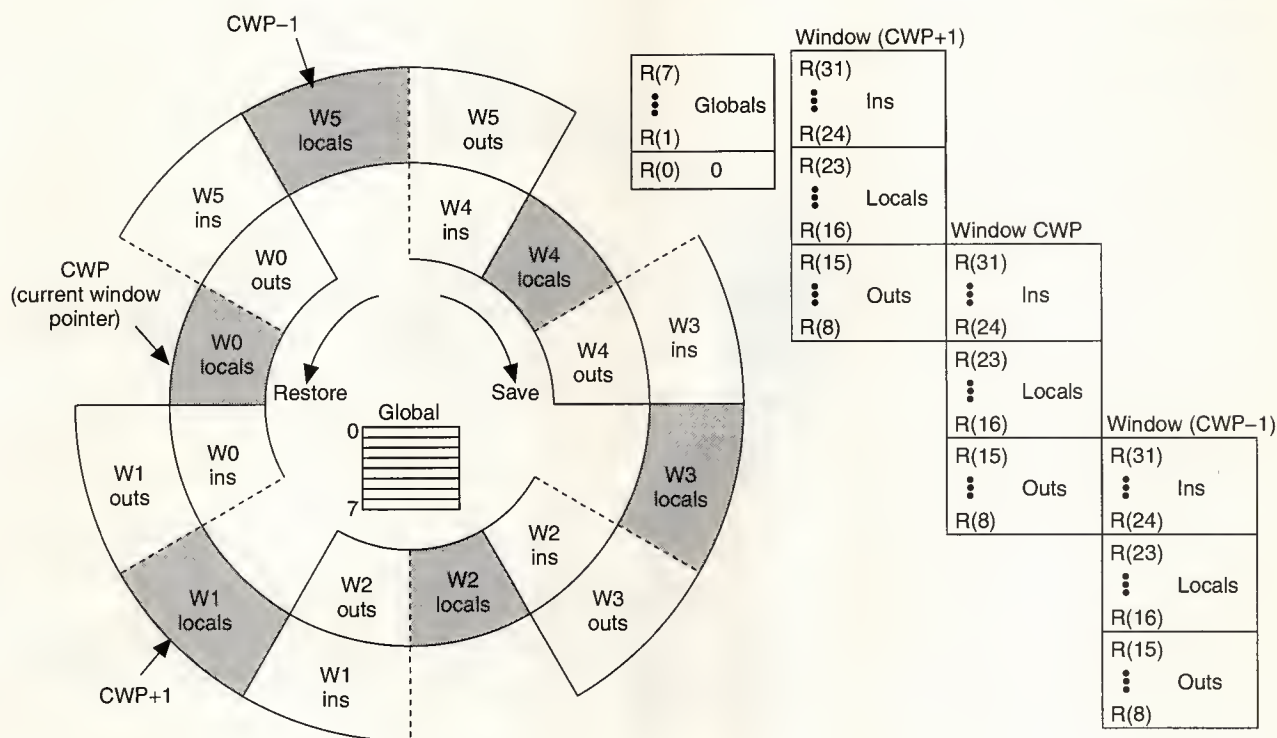


Figure 1. Register windows.

physical registers per window). It also has a set of eight global registers that are always active, regardless of which window is selected. This approach means that an implementation may have anywhere from 40 (two sets + eight global) to 520 (32 sets + eight global) registers.

Figure 1 demonstrates the circular arrangement of a register window with six window sets. Two instructions rotate the window: Save and Restore. They manipulate the current window pointer (CWP) to open a window and save the current one (when calling a subroutine) or to close the current window and restore the previous window (when returning from a subroutine). If too many levels of Save or Restore occur, the system generates an overflow or underflow trap and allows the system software to save or restore the register file to/from memory.

Eight registers always overlap between two windows, allowing a straightforward communication between subroutines. A program wanting to pass arguments to a subroutine will put the data in its current out registers and then perform a Save and a Call to the subroutine. The subroutine acknowledges the arguments now in its in registers. Upon return, the subroutine may modify its in registers, which become the out registers of the calling procedure.

Since P1754 defines a 32-bit architecture, the number of register windows is limited by the size of the window invalid mask (WIM) register. This register is 32 bits wide with one bit per window; it determines window overflow and underflow conditions. Other unique aspects of this architecture include the fairly rich set of memory addressing modes and the support of

tagged data structures and supporting instructions.

P1754 does not define an interface for this architecture, thus leaving it to the implementers and allowing them to make this decision depending on their needs. With an independent interface architecture, the specifications may require building devices in many different speeds and performance grades. For a low-cost implementation, a developer may decide to include a simple cache and a single 32-bit bus. For a high-performance implementation, a developer may decide to design a superscalar device with two different 64-bit buses for data and instructions. Other specifications may require gallium arsenide technology over CMOS, while even others may designate ECL. With this flexibility, it is almost guaranteed that companies will be interested in developing multiple-

performance workstations that can take advantage of binary compatibility. See Figure 2 and Table 1.

What does it mean to have an open microprocessor architecture standard?

- **Software compatibility.** By following a simple set of rules, software developers can ensure the portability of their software between all P1754-compliant systems, be it a high-performance superscalar micro or a low-cost desktop workstation. This set of rules, or Application Binary Standard (ABS), outlines the features of P1754 that may prevent portability (supervisor instructions and implementation-dependent extensions) and is not recommended for portable code. Software developers targeting the workstation market can now easily shrink-wrap their software packages and sell them in retail stores as other developers do for PCs.

- **Ease of implementation.** Developers of P1754 implementations can focus all their energy on important issues, such as developing better technologies, higher integration, better interfaces, and more powerful memory models and cache subsystems. Hardware engineers can devote their time to developing the internal data path, ALUs, and other logic blocks, without worrying about defining the architecture.

Other implementation challenges may include caches, direct DRAM interfaces, high-performance floating-point units, and unique bus interfaces to enhance the performance or to lower the overall cost. Packages and power dissipation will become more important as the popularity of this standard increases and will help system developers to build real-world laptop and notebook systems.

- **System compatibility.** System

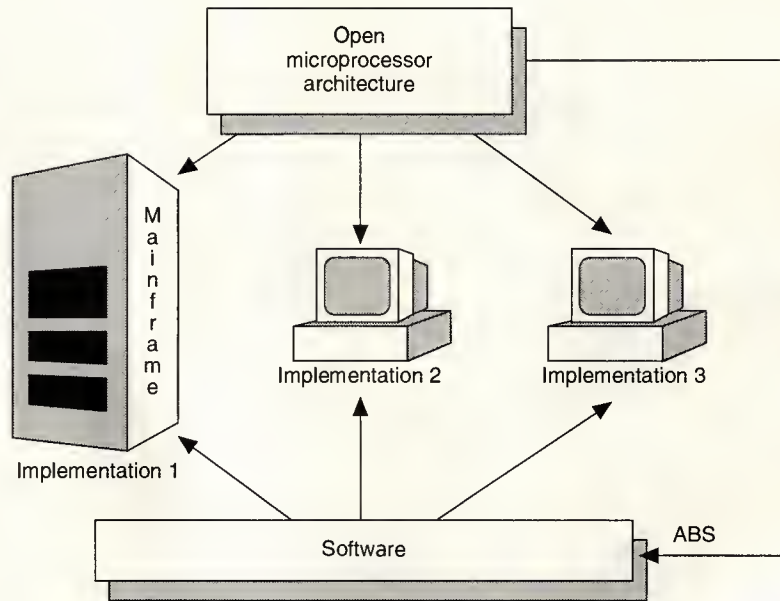


Figure 2. P1754: One standard, multiple implementations, and one-time software development.

Table 1. Offerings of P1754 and other architectures.

Features	68000, 88000, i860	P1754
Interface/bus flexibility	Very limited	Many
Architectural extensions	None	Many
Scalable architecture	No	Yes
Number of implementers	One	Many
Performance grades	Few*	Many

*Mainly different clock speeds

developers will be able choose between a wide variety of P1754 implementations, and can select the right parts for the overall system performance. They won't have to invest large amounts of money and manpower to rewrite software with the wide variety of

portable application available. Using open bus architectures like SBus or Futurebus will even allow developers to swap peripherals and extensions among all workstations. System developers can now enter the workstation market and compete with others

based purely on the performance of their workstations and not on the number of available software packages.

Look back at the model of the IBM PC. It took several years until the standard emerged. Can't we see many more advantages in a widely accepted standard? Having a strong base of supporters for P1754 can make it "the" standard for the 90s. We are already seeing the emergence of open system buses like the SBus, MBus, Futurebus+, and others, which will contribute to the development of open systems and evolve into a large base of extensions for users.

In June 1991 JTC1-SC26 (the Joint Technical Community on Information Technology) approved sponsorship of a working group to make P1754 an international standard. Balloting for P1754 occurred in September 1991; we'll have a standard by the end of 1991 or early 1992.

Rudolf Usselmann, a senior architect at Sparc International in Menlo Park, California, develops extensions to the Sparc architecture and related products such as the SBus and MBus and performs compliance testing on Sparc processors and peripherals. He has worked for Lattice Semiconductor Corporation, LSI Logic Corporation, and Sun Microsystems, and has owned and operated his own computer systems company in Germany.

Usselmann holds a Diploma in computer science and electrical engineering from the University of Hamburg.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 192 Medium 193 High 194



Richard H. Stern

Law Offices of Richard H. Stern

1300 19th Street NW, Suite 400

Washington, DC 20036

Fraud on the Copyright Office

Judge Hatter had computer program copyright owners worried for a while after he held all of Ashton-Tate's dBase copyrights invalid because Ashton-Tate had committed fraud on the United States Copyright Office. (See *Micro Law*, June 1991.) Since his December 1990 ruling, however, he changed his mind and decided that maybe Ashton-Tate wasn't so fraudulent after all. The dBase copyrights have therefore been resurrected, for the time being at least.

Nonetheless, his original ruling serves as a warning that it is risky to be careless about what you put on applications filed in the Copyright Office for registration of copyrights and mask works. Ashton-Tate's problem concerned its failure to disclose what parts of the work it submitted for registration came from prior sources. That probably remains the most likely source of a fraud problem, and should be considered first.

Application forms for registration of computer programs (Copyright Office form TX) and mask works (Copyright Office form MW) require the applicant to state whether any of the material sought to be registered comes from a preexisting source, and if so to identify it. The form for computer programs directs the applicant to "identify any preexisting work or works that this work is based on or incorporates" (Space 6a). The form for mask works merely directs the applicant to "describe the new, original contribution in the

mask work" (Space 8). But it mentions that mask works generally contain old material that is not protectible in itself, and directs the applicant to point out the new, protectible material furnishing the basis of the claim of mask work rights.

It has been a common, careless practice to leave Space 6a of the copyright form blank and to write in Space 8 of the mask work form (and in a similar space in the copyright form) some phrase such as "entire work." But often, that is not true, or at least it somewhat fudges the truth.

For example, in one now-pending case involving a chip design, the chip incorporated a large module from a prior chip that the applicant had marketed; the applicant failed to mention that fact. When the applicant sued a competitor, it claimed fraud on the Copyright Office. It may well be that the Copyright Office would have issued a registration on the chip, if full disclosure had been made in the first place, since there may have been protectible material in the part of the later chip *not* taken from the earlier chip. Yet, the information about the earlier chip was omitted in the face of an express directive to distinguish the new and old parts of the chip. What is worse, when the owner of the mask work sued its competitor, it took the position that the whole chip had been copied and infringed, even though a major fraction of it was the same as the earlier chip (which was in the public domain).

Whether an applicant that does that should lose all rights or just be slapped on the wrist is something the courts will have to decide. But if you are the applicant, why get involved in this kind of mess? The cost-benefit ratio is unfavorable.

A check list of what to be careful about in filling out Copyright Office registration forms should include several other items that can later cause trouble, if incorrectly stated:

- **Name of author.** The author of a computer program is the person who wrote the code, unless that person was an employee. A consultant or independent contractor who wrote the code is not an

employee, and is therefore the author. What is more, unless a nonemployee author signs a written assignment of the copyright in favor of the client, the author owns the copyright. Further, if someone else writes the code, thinking up none of the following makes you the author: having the basic idea of the program, creating its algorithms, determining what functions or tasks it will perform.

- **Year of creation, date of first publication.** It is easy to get these dates wrong, particularly when the program is written outside the United States. Language problems frequently result in mistakes.

When there are several versions or iterations of a program, confusion may occur over which one should be the basis of the registration. The same thing occurs with chips. Yet, even an innocent error about date is likely to lead to an infringer's claiming that the applicant committed fraud on the Copyright Office in procuring the registration. I was involved in one case where that led to a blatant infringer's staving off a preliminary injunction for a year, because the mistake created doubt about the copyright owner's case. The result was nearly complete destruction of the copyright owner's business.

Much, or even most, of the time charges of fraud on the Copyright Office are just a smoke screen that infringers create to hide their piracy. But some of the time, applicants really do try to hide the facts to deceive the Copyright Office and to dupe the public into paying tribute where it is undeserved. Sometimes, the tactic succeeds in slowing down competitors or scaring them off.

To learn into which category the Ashton-Tate case falls is something we will have to discover later. Whatever the answer is, however, it is clear that you do not want to be dragged into such a controversy if you can avoid it. It therefore pays to take care to write down correct answers to the questions on Copyright Office forms.

IEEE Computer Society Press

NEW RELEASES

1991 IEEE WORKSHOP ON VISUAL MOTION

The goal of this workshop is to assemble leaders and innovators in visual motion research and to promote vigorous debate on key issues in this field. The proceedings of the *IEEE Workshop on Visual Motion* is comprised of 46 papers on subjects such as image-flow, 3D motion and structure, optical flow, rigid motion, ego-motion, superposition, active vision, and statistical estimation.

Sections: Structure and Motion from Extended Sequences, Analysis of Image Flow, Combined Motion and Stereo, Models of Human and Biological Vision, Recovery of Ego-Motion, Analysis of Multiple Motion, Robotics and Navigation, Object Shape and Segmentation from Motion, Non-Rigid Motion.

360 PAGES. OCTOBER 1991. SOFTBOUND. ISBN 0-8186-2153-2.
CATALOG # 2153 — \$70.00 MEMBERS \$35.00



ORDER TOLL-FREE
1-800-CS-BOOKS
OR FAX (714) 821-4010

in California call (714) 821-8380



Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 177 Medium 178 High 179

Software Report



A glimpse of the future

What follows is a potpourri of recent items that were of interest to me, sometimes because of their relation to a report I wrote in the past, or because they provide a glimpse of the future. I plan to follow up on a few of these in more detail, but some readers might find these previews useful. I apologize for the random nature of these short notes.

David K. Kahaner

US Office of Naval

Research, Far East

kahaner@cs.titech.ac.jp

Sales: LCDs, workstations

LCD sales are booming; Sharp, Sanyo, Matsushita, Toshiba, and others project very large sales boosts. Color liquid crystal TVs and projectors are expected to reach 1.8 million and 150,000 units respectively. LCDs are also being developed by Alps Electric, NEC, Mitsubishi, Asahi Glass, and so on. This key technology will play an important role in future computers. And, of course, Japan produces the best (largest and brightest) LCDs. Also, consumer electronics technology feeds into computers, and firms known in the West for their consumer electronics (Sony, Sharp, Seiko) are also heavily into computer activities.

Seiko plans an LCD research and development center in Italy this year with 20 to 30 researchers. Seiko produces 10-inch PC LCDs. Omron also has an LCD with 320 × 160 dots, a contrast ratio of 8:1, and view angles of 45 degrees in all directions. It plans to produce an eight-color panel. Matsuzaki Vacuum plans a \$75-million US plant designed to produce 100,000 ten-inch panels each month.

In 1990, engineering workstation sales jumped 53 percent to about 98,000 units. Of these, Sun Microsystems produces about 25 percent, Sony 15 percent, and NEC 10 percent. X-window terminals are estimated at about 5,000 units. By the

end of 1991, another 30 percent increase is projected, with X terminals doubling to about 10,000 units.

Toshiba has revised upward by 100 percent its sales goals for laptop workstations. The new LCD production plant allows Toshiba to produce about 1,000 units/month.

Fujitsu plans a complete remodel of some of its workstation series to be compatible with Sun.

Microcomputers

A 1990 edition of *Techno Japan* contains two lengthy articles surveying microcomputers that were designed and built entirely in Japan.¹ It describes in detail products from NEC, Hitachi, TRON, and so on.

One of the most interesting remarks in the article is a claim that Japan has not caught up with the US in this technology. The only area in which the authors claim that Japan is on a par with the US is in DRAM production.

One article states,

Given then that the Japanese industry is leading in the manufacture of DRAMs, one should note that the most important technology in the field is not that of DRAMs but rather of MPUs. The world market of MPUs is now dominated by Motorola and Intel, and all Japanese computer manufacturers employ these chips in their machines. Furthermore, US manufacturers have amassed a considerable amount of MPU-related software upon which Japanese manufactures rely heavily. In addition, license fees must be paid to US firms by Japanese companies manufacturing semiconductor products,

continued on p. 79



New Products

Send announcements of new microcomputer and microprocessor products to
Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

Joe Hootman

University of
North Dakota

Computers/systems

Desktop performs at 21 Specmarks

The Tempest-compliant Sparcstation 2GX graphics workstation targets RISC/Unix government users and contractors requiring secure hardware capabilities. Along with 16 Mbytes of memory and accelerated 2D graphics, the desktop features a key-lockable front door for controlled access. A 40-MHz CPU offers 21-Specmark, 28.5-MIPS, and 4.2-Mflops performance ratings.

The system is binary compatible with the company's Sparc-based computers. The standard Tempest system includes a 19-inch color monitor, 8-bit 2D/3D graphics (using the GX graphics accelerator), a 3.5-inch DOS-compatible floppy disk drive, keyboard, and mouse. *Sun Microsystems, Inc.*; from \$29,995.

Reader Service No. 10

Carry a pen-based notepad

Field professionals and technicians can print directly on a digitized screen with a cordless pen to input data for processing and analysis when using the NCR 3125. The 3.9-lb. notepad recognizes upper- and lowercase block print letters and is said to adapt to multiple handwriting styles.

The 386 SL-based NCR 3125 supports up to 16 Kbytes of cache memory and MS-DOS, Microsoft Windows for Pen Computing, and Go Corporation's Pen Point operating systems. Standard 4-Mbyte memory can be expanded by users to 20 Mbytes in 2- and 4-Mbyte increments of DRAM or EEPROM in SIMMs or IC cards. *NCR Corporation*; \$4,765 including charger, carrying case, and stylus.

Reader Service No. 12

Color laptop features the 486

The 16.8-lb. Pro Speed 486SX/C laptop com-

puter contains a 10-inch, Super VGA, thin-film transistor, active-matrix color screen designed for high-end users. Support for 256 colors, 640 × 480 resolution, 120-Mbyte hard drive, 2-Mbyte standard memory (expandable to 20 Mbytes), and a built-in 32-bit EISA slot help the Pro Speed provide optimal expansion for networking, imaging, and engineering applications. The laptop runs Windows 3.0 and MS-DOS 5.0 as operating environments. *NEC Technologies, Inc.*; \$8,999.

Reader Service No. 11



NEC Pro Speed 486SX/C laptop

Turnkey storage system

A storage server designed for RISC System/6000 workstations runs visualization, simulation, and high-speed data acquisition and analysis applications. With up to forty 5.75-in. ESDI hard-disk drives working together, the RS/6000 RAID

storage server delivers 18 Mbytes/s sustained data transfer rates and up to 43.2 Gbytes of data storage. *Maximum Strategy*; from \$112,750.

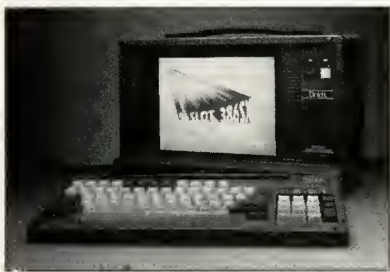
Reader Service No. 13

Portable offers five internal expansion slots

For added functionality, a 20-MHz 386SX portable computer offers mobile users five uncommitted, full-size ISA slots. The P.A.C. SX-20C includes a 16-Mbyte system DRAM, 64-Kbyte cache SRAM, and a standard red gas plasma display with 16 shades of gray scale at 640 × 480 VGA resolution. A standard P.A.C. SX-20C configuration also includes a 2-Mbyte RAM; 40-Mbyte hard disk drive; and 1.44-Mbyte, 3.5-in. floppy disk drive.

For a full-color, CRT type of display, the company offers an optional active matrix thin-film transistor, LCD flat panel with 24,389 colors. *Dolch Computer Systems*; from \$5,995.

Reader Service No. 14



Dolch Computer Systems P.A.C. SX-20C

Slide-in components

Super Gen incorporates copper/nickel-coated plastic, plug-in memory, processor, video cartridges, slide-in drives, and slide-on modules in a 486 system designed for limited-experience users. The workstation/server family runs on the CTOS operating system, which supports the Presentation Manager graphical user interface and DOS Protected Mode Interface. *Unisys Corporation*; from \$2,995.

Reader Service No. 15

Communications

Access mainframes from the Macintosh

An enhanced version of the Macimalan LAN allows Macintosh systems to communicate with IBM mainframes. Macimalan combines Distributed Functional Terminal, IEEE Std 802.2, and Synchronous Data Link Control software. With Macimalan, users select either remote or local connections to the mainframe for up to 128 concurrent users on Apple Talk LANs using Local Talk, Token Ring, or Ethernet media.

The DFT gateway component distributes up to 20 host sessions across the LAN. The 802.2 gateway provides users in the same building with local access to the mainframe over a Token Ring backbone LAN to support up to 128 host communications sessions. The SDLC component supports a remote line to the mainframe. *Digital Communications Associates*; from \$1,495.

Reader Service No. 16

System 7 connections

Apple Macintosh users can access various host systems with the mxConnect software via asynchronous, VIP synchronous, VIP server, TCP/IP, LAT, X.25, or Communications Toolbox protocols. The System 7.0 mxConnect includes support for Kermit, Xmodem, Ymodem, Zmodem, non-protocol, mxFTF, and Communications Toolbox file transfer protocols. The package requires 1 Mbyte of memory. *Cambridge Computer Corporation*; \$295.

Reader Service No. 17

Token Ring adapters

Five 8- or 16-bit Token Ring adapters ease connections in IBM PC XTs, ATs, or PS/2 Micro Channel computers. In addition, the company offers four boot ROMs to load remote programs on Novell- and Net BIOS-based LANs. *Tiara Computer Systems*.

Reader Service No. 18

Modem supports MAP LANs

Manufacturing Automation Protocol users can interconnect machines in factory environments with a baseband (no carrier frequency) modem called the MHW11005. Based on the MC68194 carrierband modem, the MHW11005 operates at a 5-Mbps data rate, uses frequency shift keying, and can be inserted into a motherboard. Operating in temperatures from 0°C to +70°C, the PCB module supports equipment interconnected via coaxial cable for digital information exchange purposes. *Motorola*; \$250 (500s).

Reader Service No. 19

Transfer data over 5,000 meters

The Roadrunner series of IEEE-488 bus extenders transfers data over 5,000 meters using a proprietary data transmission protocol. Models 4889A and 4889B transmit at 300 Kbytes/s and offer error detection and correction capabilities. According to the company, the nondegrading data rate does not appreciably affect system response time.

Each 115-VAC extender offers standard coaxial and fiber-optical interfaces. *ICS Electronics*; from \$1,695, 2 weeks ARO.

Reader Service No. 20

Communication software helps novices, experts

Pere Line 3.0 is a commercial communications program with Windows and memory-swapping features that let experts explore numerous configuration operations. In contrast, novices may let Pere Line 3.0 work automatically once they have used the SAA-style pop-up menus.

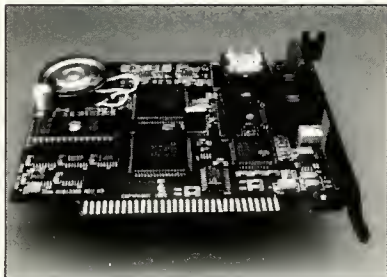
The package supports 45 modems and ISDN adapters, enables another application to run while files are being transferred or downloaded, and operates on PC/AT/XT or PS/2 compatibles. *Pere Line Data Systems*; \$49.95 plus \$6 shipping and handling.

Reader Service No. 21

PC fax modem

An internal modem for PCs combines a 2,400-bps data modem and 9,600-bps Quick Link II send/receive fax capabilities. The PM2400FX96 supports the Hayes AT command set and lets users receive a fax while using the computer for other purposes. Users can send multiple files with one phone call and send a fax directly from the DOS prompt when using the modem. The PM2400FX96 comes with a five-year performance warranty that covers parts, labor, and factory repair or replacement. *Practical Peripherals, Inc.*; \$209.

Reader Service No. 22



Practical Peripherals fax modem

Transceivers reduce costs

A 2.5-ounce transceiver with a two-year warranty reduces Ethernet cabling costs by replacing AUI drop cables with inexpensive unshielded twisted-pair cables. Status indicators on the AT-110T transceiver constantly monitor DTE power and valid links, while a loop-back function emulates coaxial media to determine whether a media access unit is connected. *Allied Telesis Inc.*; \$79.95 (volume discounts).

Reader Service No. 23

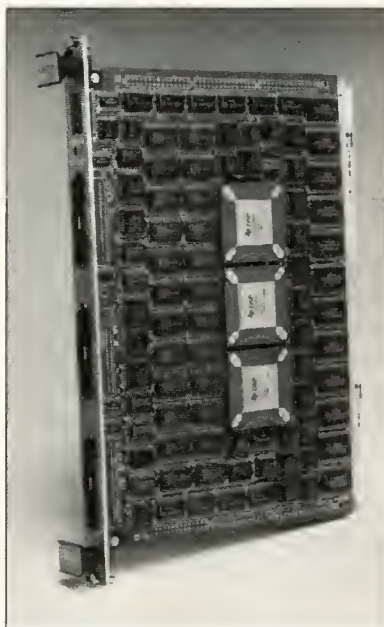
DSP hardware/software

VMEbus DSP moves data at 200 Mbytes/s

The DSP-3A multiprocessor board includes three connected, 40-MHz TMS320C30A devices to support fast parallel or pipelined DSP operations.

A set of graphical object-oriented programming tools available for the DSP-3A speeds application development, and a graphical software tool lets tasks be wired together pictorially on a Sun or X-Window workstation. The DSP-3A fits in a 6U slot of a standard VMEbus card cage. *PC/M Inc.*; \$17,779 each; 45 days ARO.

Reader Service No. 24



The PC/M DSP-3A board

DSP for the PC

PC Data Master 3.0 is a signal processing system for 512-Kbyte IBM PCs and compatibles. The system combines graphics, data-sampling, test data generation, and data file math routines with DSP and general-purpose utilities. Users can integrate their own data analysis and graphics functions using various language compilers or MS-DOS 3.0 assemblers.

PC Data Master runs best with a hard disk, 1.4-Mbyte on-line disk space, and 640-Kbyte RAM. *Durham Technical Images*; \$185 plus \$95 for an academic site license.

Reader Service No. 25

System controls seven SCSIs

Designed for use in PC AT-bus computers, the MP3210 system uses a DSP3210 32-bit, floating-point chip and a DT-Connect interface to provide digital and audio I/O and real-time video signal transfer. Capable of controlling seven SCSI devices, the MP3210 speeds multiprocessing, multitasking, and applications development tasks. The system's DSP56ADC16 sigma-delta oversampling converters offer analog I/O with a signal-to-noise ratio of 90 dB. Two 16-bit analog channels provide concurrent outputs at sample rates up to 50 kHz per channel and include digital oversampling reconstruction filters. *Ariel Corporation*; \$4,995.

Reader Service No. 26

Sampling converters

Two 12-bit sampling monolithic converters plug into most ADC574 sockets without system modifications. With switched capacitor array CMOS structures, the ADS574 and ADS774 feature internal sampling and one +5V power supply operation. Users can control the sampling function to eliminate the need for external sample/holds in most designs.

ADS574's maximum throughput time for 12-bit conversion is 25 μ s, including acquisition; ADS774 converts and acquires data in 8.5 μ s. Both converters come in 0.3- or 0.6-inch-wide, 28-pin plastic or side-brazed hermetic ceramic DIPs, 28-pin SOICs, and in die form. *Burr-Brown Corporation*; from \$14.15 (OEMs, 100s).

Reader Service No. 27

Embedded antialiasing filters

The DT3831 series of 12-bit, 50-kHz or 250-kHz, one-slot AT-compatible data acquisition boards include embedded antialiasing filters and the company's Real-Time Error Prevention circuit. Front-ending each analog input with a 4-pole Butterworth filter on one board reduces noise inherent in alternative two-board solutions while

maintaining accuracy within the filter's passband. The RTEP circuit adds on-the-fly offset calibration of channel, range, and gain value combinations.

Each board contains a driver, toolkit, and a Gallery demonstration package. *Data Translation; from \$3,695 each.*

Reader Service No. 28



Data Translation DT3831 data boards

SBUS TMS320 board

An SBUS board built around the TMS320C30 DSP provides a development platform and real-time data acquisition capability for the Sparcstation. With 512 Kwords of SRAM and dual-port RAM, the board adds speed, arithmetic commands, and address modes for real-time I/O to Sparc's central processor performance. The company's 16-bit DSP-Link expansion bus permits the SBUS board to be connected to standard interfaces and provides communication between SBUS C30 boards. *Spectrum Signal Processing Inc.; \$4,595 each; optional analog I/O modules, \$995 each.*

Reader Service No. 29

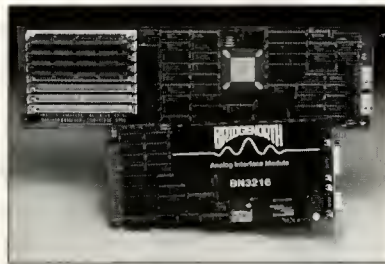
Integrated PC system

A DSP development and data acquisition system contains the BN3000 32-bit, floating-point DSP processor board, and the BN3216 two-channel, 16-bit analog interface module.

Together, the board and module provide an integrated 33-Mflops signal-processing system in a PC. Alone, the BN3000 system analyzes signals, performs high-speed computation in a PC, and can be expanded with a 32-bit parallel I/O expansion bus through two

TMS320C30 serial ports. *Bridgenorth Signal Processing, Inc.; \$2,995 (BN3000), \$995 (BN3216).*

Reader Service No. 30



Bridgenorth Signal Processing system

A/D components

RS485 drivers draw 200 μ A

CMOS/Schottky communications devices called the LTC486 and LTC487 feature 28-ns driver propagation delays with 5-ns skew. Both ICs incorporate a thermal shut-down protection feature and come in 16-pin molded DIP and 16-lead, wide-body SOIC packages. Each IC supports data-transmission rates up to 10 Mbps, operates from one 5V supply, and draws 200 μ A of supply current. *Linear Technology Corporation; \$3.70 (samples, 100s).*

Reader Service No. 31

SRAMs in ZIPs

Two JEDEC, 2-Mbit fast SRAMs in zag in-line packages promise 15- and 20-ns access times. The MCM3264 contains eight MCM6209 64K \times 4 fast static RAMs, while the MCM8256 uses eight MCM6207 256 \times 1 fast SRAMs. *Motorola MOS Memory Products Division; from \$195 (100s).*

Reader Service No. 32

Laser printer controller

High-volume users, OEMs, and VARs can use the LC-8000 laser printer controller to produce text and graphics output for CAD/CAM, law, insurance, and finance applications. An input

buffer allows information to be fed into the printer at 2 Mbytes/s; separate sections of the controller draw characters, manage a font library, erase the page image, and print. Text and graphics can be output at the speed of a 20- or 50-ppm print engine. *Advanced Technologies International; \$3,750, volume discounts available.*

Reader Service No. 33

VGA notebook controller

The 160-pin, quad flat pack CL-GD6410 controller lets a VGA subsystem be implemented with five chips that occupy 4 square inches of space. Features include 64 gray shades on a monochrome LCD, a 512-color active-matrix LCD, and simultaneous operation of the LCD panel and an analog CRT monitor. Also required to create the VGA/CRT subsystem are two DRAMs for video memory, a clock synthesizer, and a 64K \times 4 frame accelerator memory, if a dual-panel LCD is used. *Cirrus Logic, Inc.; \$50 each (samples).*

Reader Service No. 34

Four-Mbyte floppy-disk controller

The FDC37C65C+ floppy-disk controller incorporates support for 4-Mbyte floppy-disk drives and most drive formats. The "+" designation represents the addition of a 16-byte FIFO and vertical recording format mode to the company's FDC37C65C. *Standard Microsystems Corporation; \$6.31 (50s).*

Reader Service No. 35

EPROM adapters available

Users wishing to replace an older IC with a more-available type can upgrade boards with two EPROM adapters. The adapters contain Correct-A-Chip technology and rerouting circuitry to switch pins.

A 28-pin EPROM adapter, the 2564, switches pins 1 to 28, 2 to 23, 20 to 22, 22 to 27, 23 to 20, and 27 to 28. A second 28-pin adapter, the 2532-2732A, switches pins 18 to 21 and 21 to 18.

Aries Electronics, Inc.; \$10.68 (500s).

Reader Service No. 36

Op amps improved

According to the company, its LT1124 dual and LT1125 quad operational amplifiers outperform the OP27, the OP270 dual, and OP470 quad op amps they were designed to replace. For example, improvement ranges from a factor of 1.6 to 3.7 in slew rate and greater bandwidth, lower bias and offset currents, and higher gain than the OP27. Applications include instrumentation amplifiers, active filters, low-noise signal processing, microvolt accuracy threshold detection, and infrared detectors. *Linear Technology Corporation; from \$3.65 (100s), from \$7.05 (military temperature grades).*

Reader Service No. 37

Boards/test equipment

Is there a 500-user PC in your future?

An intelligent I/O controller supports high-speed, Unix-based multiuser systems with up to 512 ports when used with four host cards. One RIO host card supports up to 128 ports at sustained throughput levels of 57.6-Kbps full duplex.

According to the company, RIO transmits and receives data on all 512 ports simultaneously with no degradation. RIO's Inmos T225-based remote terminal adapters link to a host, such as a 386 or 486 PC with Unix or a RISC-based Unix workstation, through four 10-Mbps data channels, which are connected to the adapters placed up to 250 feet from the host. *Specialix Inc.; \$795 (cards), \$800 (adapters).*

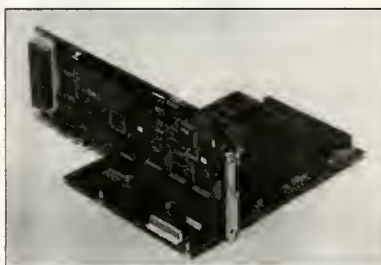
Reader Service No. 38

EISA backplanes with bus mastering

Two passive-bus EISA connector boards feature a bus master connector as an upward migration path from 8- or 16-bit ISA cards. The eight-slot

QC1118 and the 12-slot QC1119 each contain one slot with a proprietary EISAEX connector to control up to six EISA bus master slots. Both boards provide IBM PC AT keyboard interface, speaker driver, and external reset input functions. Designed for OEMs and system integrators, the backplanes mount in most cabinets that accommodate an AT motherboard. *MNC International; from \$295 (evaluation units).*

Reader Service No. 39



MNC International EISA boards

100-Mflops i860 accelerator

Developed around the 64-bit Intel i860/XP, the DASH!860/50 application accelerator offers 100-Mflops and 50-MIPS performance when installed into a 486/386 or 286 PC with 1 Mbyte of memory. The 50-MHz add-in uses an expansion interface to transfer data at 160 Mbytes/s and supports vectorizers, math libraries, graphics, and neural network packages. A memory manager supports zero-wait-state, pipelined memory access; the board's 8-Mbyte DRAM can be expanded to 32 Mbytes. *Myriad Solutions Ltd.*

Reader Service No. 40

Single-boards offer versatility

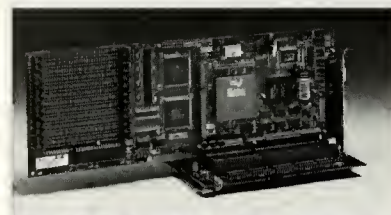
The I386 single-board computer was designed around the Intel 80386DX CPU to run at 25 or 33 MHz and features a 32- or 128-Kbyte cache with up to 32 Mbytes of 70-ns SIMM DRAM. The I386 also incorporates either an Intel 80387DX coprocessor or a Weitek 3167 numeric data processor.

A second computer, the I486, contains an internal floating-point unit, an 8-Kbyte internal cache, and either the

25- or 33-MHz Intel 80486DX combined with the 496 Super Cache.

Both boards include a hardware/software programmable watch-dog timer; a built-in, extended set-up program for programmable I/O and bus speeds; and buffered drivers capable of accommodating up to 19 additional expansion cards. *IBus PC Technologies.*

Reader Service No. 41



IBus PC Technologies boards

Imager works without frame grabber

The EDC-1000HR digitizes images into 754 × 488 eight-bit pixels without the need for a frame grabber or third-party hardware or software. The digitally controlled/digital output, monochrome camera works with IBM PCs, XTs, ATs, or equivalents through its PC-controlled interface card. Users can operate the asynchronous camera in either interlaced or noninterlaced modes, saving TIFF and PCX file images for further image processing and desktop publishing packages. The EDC-1000HR uses a frame-transfer CCD image sensor configured into 244 lines with 754 elements in each line. *Electrim Corporation; \$850.*

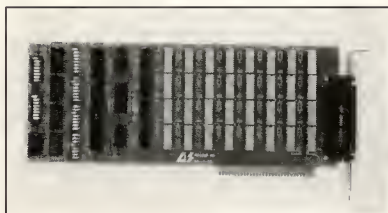
Reader Service No. 42

Solve automation problems with 32-switch card

An 8-bit computer board lets PCs select and control 32 analog or digital signals for use in laboratories, factory automation, environmental control systems, and security systems. Since there is no address limit to the number of boards that can co-reside in one PC, this board can control the connection of multiple signals through as many switch cards as a system has slots avail-

able. The 32-Switch Reed Relay Card contains reed relays that are transparent to the signals they are controlling. According to the company, signals up to 100 volts and 10 watts are unaffected while controlled by the computer. *Accusys, Inc.*; \$395.

Reader Service No. 43



Accusys, Inc. relay card

In-circuit emulator

System designers using 16- and 32-bit microprocessors can monitor system elements with the Mice-V real-time in-circuit emulators, stopping at any point to view the system for bugs.

When supporting 486SX or 487SX applications, the Mice-V-486, 486SX, and 487SX emulators provide an isolation mode so users can remove the 486 chip from the probe socket and connect logic analyzer clips to the pod to gather timing information. *Microtek International, Inc.*; from \$29,500; 4 weeks ARO; updates available.

Reader Service No. 44

Laser test source

The S795 dual laser test source for single-mode fiber optic networks combines 1,300-nm and 1,550-nm lasers into a hand-held package. Lasers in mounts couple directly with FC or ST connectors to reduce the size and cost of pigtailed lasers. When used with a typical fiber optic power meter, the S795 lets users test cable loss up to 45 dB with CW output. *Fotec Inc.*; \$3,250.

Reader Service No. 45

Power meter tests laser source

The M247 optical power meter lets factory and field users test the laser diode source used in erasable optical disk drives or laser printers. The tester uses a separate 5-mm-diameter silicon photodiode that is calibrated to measure optical power at 780- to 850-nm wavelengths with up to 20 mW of power. A 9-volt alkaline battery provides 100 hours of operation. *Fotec Inc.*; price depends on detector configuration.

Reader Service No. 46

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 189 Medium 190 High 191

Call for Papers

IEEE Micro invites authors to submit papers for three special issues:

- **Associative memories and processors in April 1992**
Submit manuscripts by 11/15/91
- **ICs for HDTV in October 1992**
Submit manuscripts by 3/15/92
- **Signal processing in December 1992**
Submit manuscripts by 5/15/92

Submit six copies of papers to:

Editor-in-Chief Dante Del Corso
Dipartimento di Elettronica
Politecnico di Torino
C.so Duca degli Abruzzi, 24
10129 Torino, Italy
e-mail: delcorso@polito.it

or

Associate EIC Ashis Khan
Mips Computer Systems, Inc.
950 DeGuigne Dr.
Sunnyvale, CA 94086
phone (408) 524-7171
e-mail: ashis@mips.com



For author guidelines, contact Krista Tague, IEEE Computer Society West Coast Office, (714) 821-8380, or fax (714) 821-4010.

Product Summary

Joe Hootman

University of North Dakota

Manufacturer	Model	Comments	R.S.#
Displays			
Qume Corporation	QM848, 857, 870 monitors	Noninterlaced color monitors support graphics applications, windowing software, and GUIs. The 14-inch QM848 and 857 are fully noninterlaced at resolutions from 640 × 480 to 1,024 × 768 and feature a 0.28-mm dot pitch. The QM870 features a 17-inch flat screen with noninterlaced display up to 1,280 × 1,024 pixels and a 0.26-mm dot pitch. From \$799.	80
Telex Communications	Magna Byte 6001	Full-color LCD computer projection panel features Macintosh and IBM capabilities equipped with VGA output. The 640×480-resolution rear-projection panel offers 30-ms or 50-ms pixel response time for creation of smooth animation without smeared images. Users may select 8, 64, or 2,197 RGB color modes with a 60 to 1 minimum contrast ratio. \$5,995.	81
Systems			
AST Research Inc.	Premium Exec option	Two options for notebook computer line include a 9,600-bps data/fax send-and-receive modem and a second RS232 serial port adapter. Premium Exec currently offers one serial, one parallel, and one external keyboard/mouse port. \$499 (modem), \$99 (adapter).	82
Digital Vision	Computer Eyes/RT	Color video frame grabber for PC-compatible computers supports multimedia, animation, remote image telecommunicating, and industrial machine vision applications. The board includes software with drop-down menus for 1/30th-second grabs and 512 × 512 resolution at 24 bits per pixel in 16 million colors. \$599.95.	83
Gespac	CVICLN1728T	Line of linear-scan CCD camera, interfaces, and software includes a 1,728-pixel resolution and produces a 6-bit digital output for 64 gray levels. A 2-Mbyte/s signal output permits the camera to be placed 100 meters from the microcomputer. Two cameras can use a CVICLN-G2 board to interface to a G-64 system running OS-9 or MS DOS. \$1,655 (camera), \$995 (interface).	84
Software			
Adobe Systems	Photoshop V. 2.0	Apple Macintosh image-processing package includes enhanced color and black-and-white image editing, direct CMYK editing, editing of selected image areas, and importing of Illustrator-compatible EPS files. Requires an SE or II with 2 Mbytes of RAM, hard-disk drive, and System 6.04 or later software. Also recommended is a 68020 CPU, color monitor, and 4 or more Mbytes of RAM. \$199 (upgrades).	85
Applied Microsystems	Code Tap 386SX	Runtime development tool debugs software programs running on Intel 886 SX microprocessors and monitors and controls execution in the target without using target memory or I/O, or requiring prior	86

Manufacturer	Model	Comments	R.S.#
		code modification. The tool chain consists of a target access probe, an RS232 communications adapter, windowed source-level debugger, Validate/Softscope III, and Pharlap ASM/Linkloc. \$5,995.	
Circuit Search	Database, V. 1.09	This dBase III/dBase III+-compatible database contains references to 13,000 articles and papers from 300 technical and scientific journals and magazines. Users locate circuits by keywords after installing the database on an IBM PC-compatible hard disk and using in conjunction with either the supplied menu-driven front end or a dBase-compatible database management program. \$375 (includes one free semiannual update).	87
Inset Systems	Hijaak 2.02	Graphics utilities include a Super VGA-screen capture program for Windows 3.0 that saves up to 256 colors simultaneously. Once saved to disk, Hijaak converts the image to supported raster formats, fax card formats, or Post Script. Also available are a 5-Kbyte TSR for DOS and a conversion program for 36 graphics file formats. Free or \$50 (upgrades), \$199 (all others).	88
National Instruments	Lab Driver for Windows	Dynamic link library operates under real, standard, and enhanced modes of the Microsoft Windows 3.0 operating system. The library controls the company's plug-in data acquisition boards for XT/AT, PS/2, and EISA PCs with programs written in programming languages that support Windows' DLLs. Low-level functions for analog, digital, and timing I/O and high-level functions for stream-to-disk and waveform generation programs control the boards.	89
Miscellaney			
Cubit	Model 7050 controller	Color graphics CRT controller communicates on an 8-bit or 16-bit STD bus to offload most of the graphics-handling load from the main system CPU. The 7050 supports up to three screens with 640 × 480 resolution, includes 4 Mbytes of video RAM, and is compatible with VGA, EGA, and monochrome monitors. \$490 each.	90
Martech	Workstation RAMs	Memory boards provide an alternative to HP Apollo 9000 series 700 workstation RAMs. The 16-Mbyte and 32-Mbyte module memory boards are available from stock for next-day delivery. \$3,360 (16 Mbytes), \$6,720 (32 Mbytes).	91
National Instruments	PC-LPM-16 I/O board	Surface-mount version helps users develop portable data acquisition systems using laptop computers. The 4.35-inch × 3.9-inch board supports applications of signal and transient analysis, data logging, switching external devices, reading the status of external digital logic, synchronizing events, generating pulses, and measuring frequency and time.	92
Specialized Systems Consultants	ANSI C reference card	Revised reference card reflects changes in the finalization of the ANSI C standard established by the X3J11 committee. The eight-page card combines examples and specifications to illustrate C and explains statement formats, functions, constants, and pre-processor commands by example. \$3.	93

Simulating a visual function

continued from p. 11

network capable of handling images with up to 64×64 resolution as illustrated in Figure 2b.

Experimental results

We performed experiments on a Sun Sparcstation 1 with the Sun OS 4.0.3 operating system, conducting SPICE simulations (XSPICE 3) with 2D test patterns and images.

Testing the PE. Our first test involved verification of the design of the modifiable thresholding unit. We used five randomly chosen T 's of image intensity to simulate possible situations in motion detection. With each given T , we then

determined the mapping voltage by Theorem 1. Using this voltage, we specified the W/L ratio by using Theorem 2. Next, we constructed a thresholding unit and simulated the processing element. Table 1 lists a set of the simulation results, with T , V_{in} , and the W/L ratio. Figure 3 illustrates the corresponding set of VTC curves, which match the design specifications. For example, when a gray-level threshold equals $T=127$, the corresponding voltage is $V_{in}=2.5V$, which is then realized in hardware by setting $(W_n/L_n)/(W_p/L_p)$ equal to 1.

Testing the 2D network. We constructed a small two-dimensional network with 6×6 PEs to form a 2D array processor. Each PE has two input pixels from test patterns 1 and 2 at the same (x,y) location. We randomly shifted each pattern to produce a time-variant image. To simulate a real situation of dynamic image processing, we added random noise

with a maximum 10-pixel variation to the test patterns and fed the test patterns into the network as two consecutive image frames. Each PE in the network operates concurrently, first performing differencing operations then thresholding. The processed outputs are collected at the output node of the network for comparison to the numerical computation. The experimental result illustrated in Figure 4 shows that the network behaves well under simulated noise conditions. One can see from Figure 4b that the network has captured the nonstationary region.

To test the 2D network with 64×64 analog PEs, we wrote a program in C to generate a SPICE program of the network with about 100,000 components. A block diagram given in Figure 5 illustrates the procedure. Note that all the processing results are compared to numerical computation.^{3,4}

For the next stage of testing we created synthetic test images. The first image frame contains two objects: a square and a triangle. The second image contains the same objects with the square moved to a new position. We selected a gray-level threshold $T=179$ and calculated the W/L ratio, which is equal to 0.1837, for each PE. Figure 6 on page 46 gives the processing result. Note that $D_1(x,y,t)$ corresponds to the triangular regions in Figure 6a and 6b, while $D_2(x,y,t)$ pro-

Table 1. Relationship of image intensity, voltage threshold, and W/L ratio.

Threshold T in image intensity	Voltage threshold (V_{in})	SPICE simulation	$(W/L)_n : (W/L)_p$ ratio
100	1.96	Matched	2.4056
127	2.50	Matched	1.0000
163	3.20	Matched	0.3164
179	3.50	Matched	0.1837

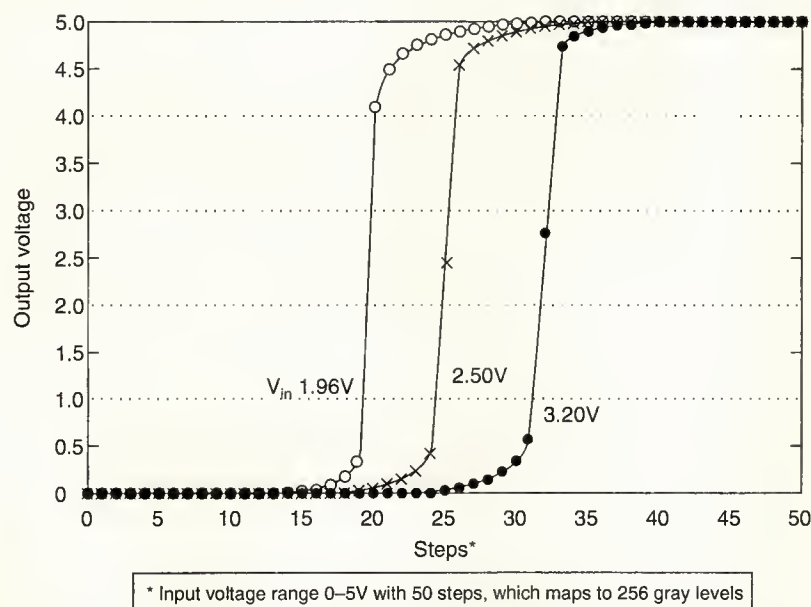


Figure 3. SPICE simulation result of the thresholding unit with different V_{in} . Note that we designed each VTC curve with a different gray-level threshold, which is implemented by defining different W/L ratios.

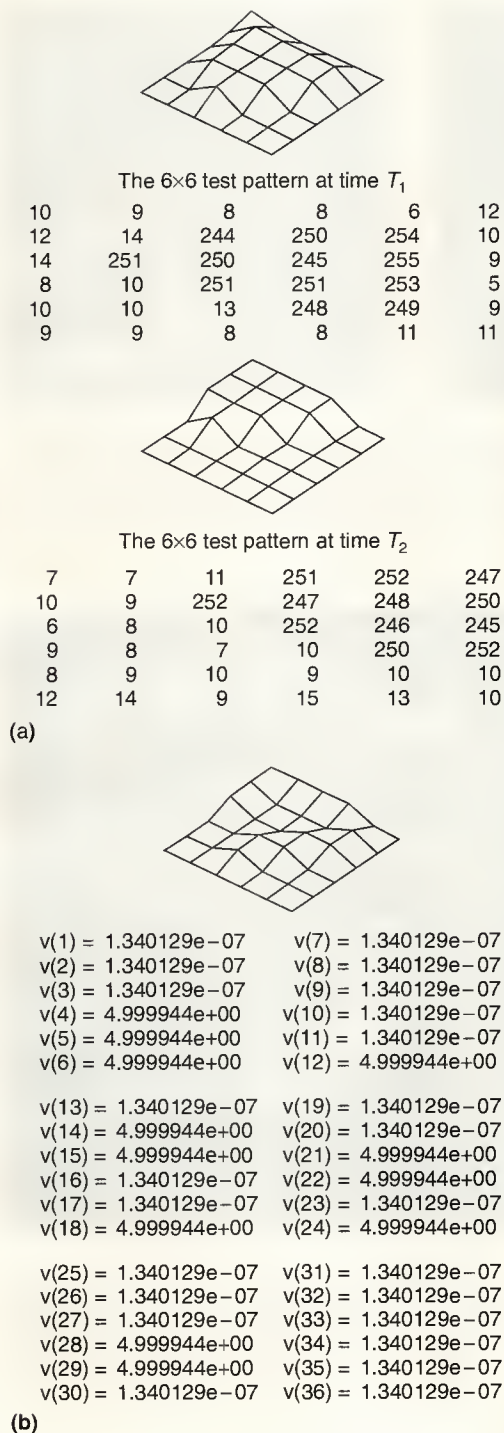


Figure 4. Two 6 x 6 test patterns, each being randomly shifted, with random noise added (a); and the SPICE simulation result (b).

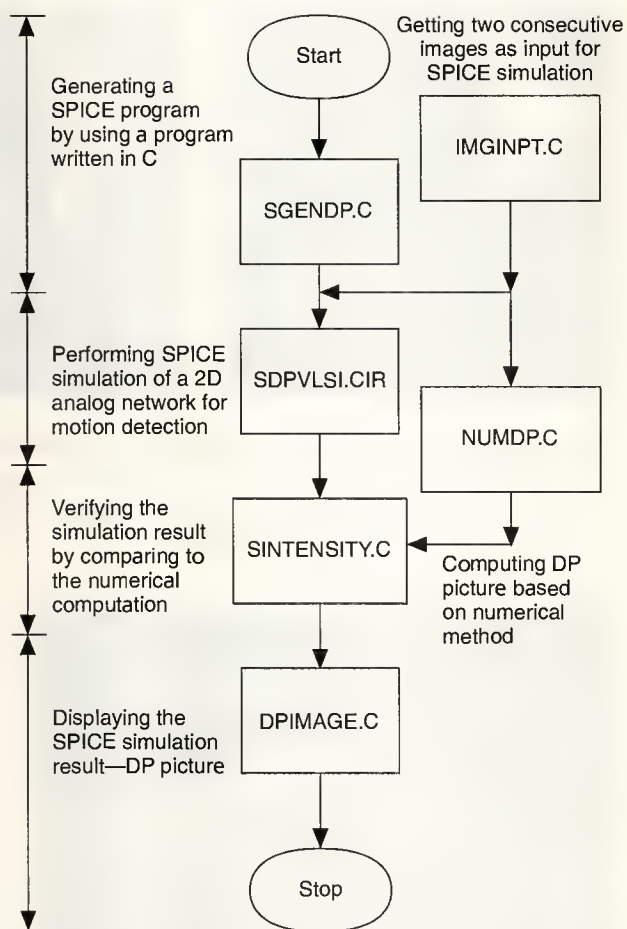


Figure 5. Block diagram showing the procedure for generating and testing a 64 x 64-PE network. DP indicates the difference picture or $D(x,y,t)$.

duced the reversed-L region in Figure 6c. We compared this image to the numerical computation and found that they matched.

Finally, using a DT2852 image grabber, we digitized a sequence of tool images from a laboratory scene. The images have 256 x 240-pixel resolution and 8-bit gray levels. We rearranged the tools on the scene and arbitrarily chose the two consecutive image frames shown in Figure 7 on page 46. We used the 64 x 64-resolution window shown in Figure 7a to circumscribe portions of the images for testing. The threshold value of image intensity is $T = 163$, and the corresponding voltage threshold is $V_{in} = 3.20V$. As a result, the processed image $D(x,y,t)$ given in Figure 7b shows the brighter region corresponding to $D_1(x,y,t)$ and the grayer region corresponding to $D_2(x,y,t)$. This result correctly picks up the nonstationary regions.

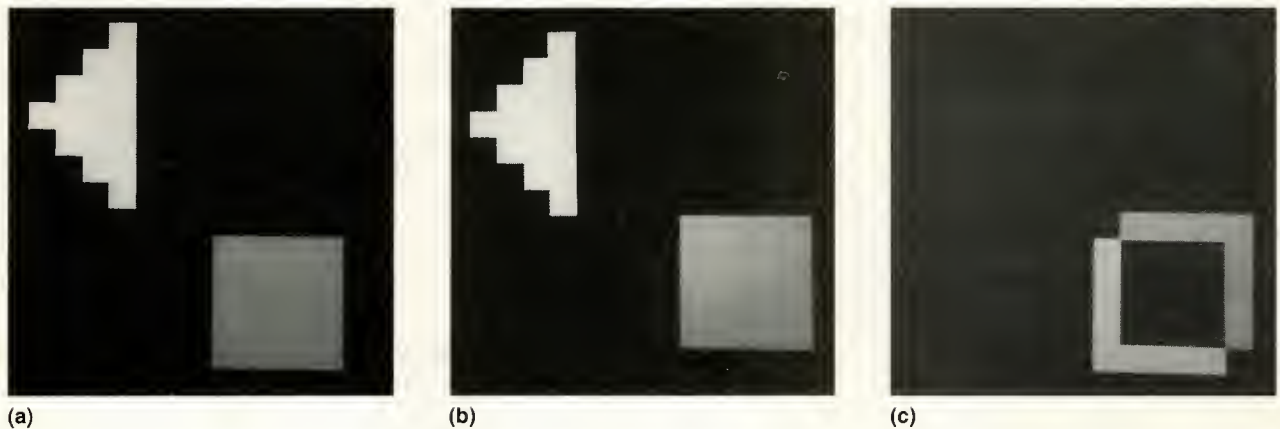


Figure 6. Testing the network: With two objects, a triangle and a square (a); with a square moved to a new position (b); and the computation result collected from the output nodes of the network (c). The nonstationary region $D_1(x,y,t)$ appears on the left, and $D_2(x,y,t)$ on the right.

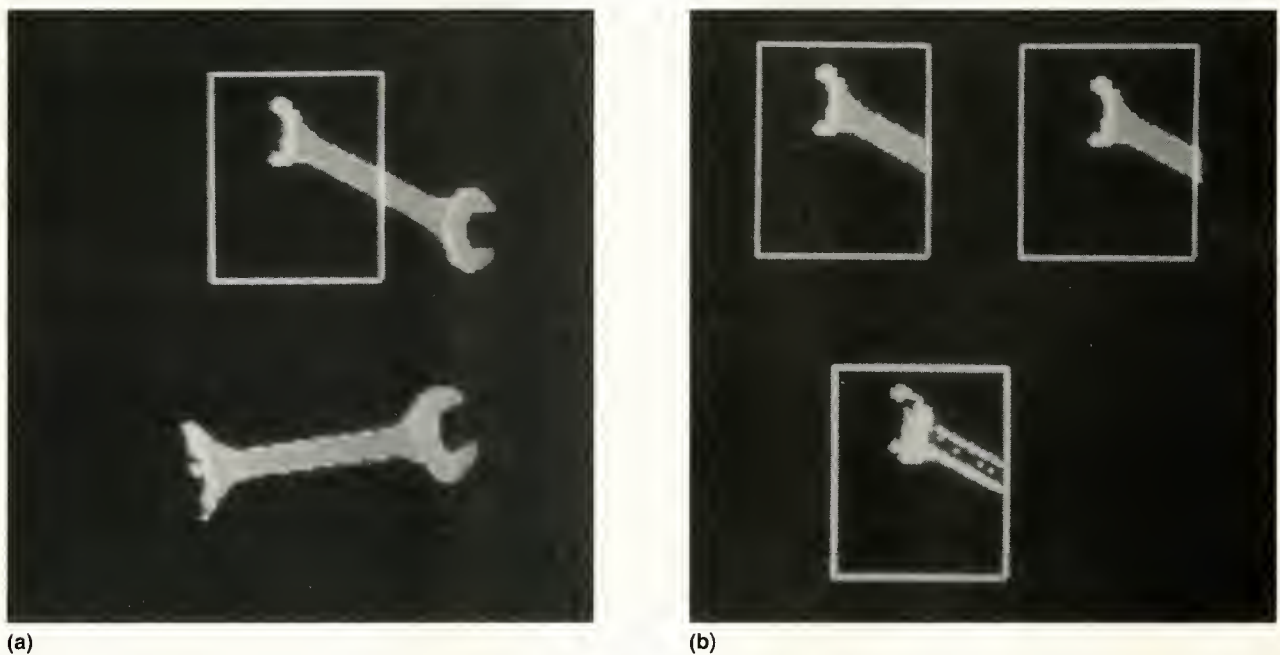


Figure 7. Laboratory images with tools displaced during acquisition: A 64×64 -resolution window circumscribes the portion to be processed (a); and two test patterns show the different positions of the tool (b). The output of the processing result from the network is displayed, clearly indicating the displacement.

OUR 2D ANALOG VLSI NETWORK simulates a function of motion perception in the human visual peripheral process. We mapped the image-processing problem to analog computing with mathematic formulation and proved theo-

rems to assist network design. Finally, we performed SPICE simulations and experiments with real laboratory images to verify the design. The network processes images in real time, matching its performance to the theoretical analysis result.

The future research of incorporating high-level visual processing is under investigation. ■

References

1. E.C. Carterette and M.P. Friedman, eds., *Handbook of Perception*, Vol. V, "Seeing," Academic Press, New York, 1975.
2. R.N. Haber and M. Hershenson, *The Psychology of Visual Perception*, 2nd ed., Rinehart and Winston Press, New York, 1980.
3. H. Li, "Two Stage Strategy for Motion Parameter Estimation in Dynamic Images," *Int'l J. Applied Artificial Intelligence*, Vol. 3, No. 4, Hemisphere Publishing Corporation, New York, 1989, pp. 427-438.
4. R. Jain, "Extraction of Motion Information from Peripheral Processes," *Digital Image Processing and Analysis*, Vol. 2, IEEE Computer Society Press, Los Alamitos, Calif., 1985, pp. 491-501.
5. M.A. Mahowald and C. Mead, "Silicon Retina," in *Analog VLSI and Neural Systems* by C. Mead, Addison Wesley, Reading, Mass., 1989.
6. J. Hutchinson et al, "Computing Motion Using Analog Binary Resistive Networks," *Computer*, Vol. 21, No. 3, Mar. 1988, pp. 52-63.
7. A. Lumsdaine, J. Wyatt, and I. Elfadel, "Nonlinear Analog Networks for Image Smoothing and Segmentation," *Proc. IEEE Int'l Symp. Circuits and Systems*, Vol. 2, 1990, pp. 987-991.
8. D.B. Tweed and T. Vilis, "The Superior Colliculus and Spatiotemporal Translation in the Saccadic System," *J. Neural Networks*, Vol. 3, 1990, pp. 75-86.
9. J.G. Taylor, "A Silicon Model of Vertebrate Retinal Processing," *J. Neural Networks*, Vol. 3, 1990, pp. 171-178.
10. D. Marr, *Vision*, W.H. Freeman Press, San Francisco, 1982.
11. M. Nagao, "Focus of Attention in the Analysis of Complex Pictures Such as Aerial Photographs," *Real-Time Parallel Computing, Image Analysis*, Morio Onoe et al., eds., Plenum Press, New York, 1981, pp. 185-191.
12. L. Uhr, "Psychological Motivation and Underlying Concepts," *Structured Computer Vision*, S. Tanimoto and A. Klinger, eds., Academic Press, New York, 1980, pp. 1-30.
13. S. Ullman, "Analysis of Visual Motion by Biological and Computer Systems," *Computer*, Vol. 14, No. 8, Aug. 1981, pp. 57-69.
14. T. Reuter, "Standards Conversion Using Motion Compensation," *J. Signal Processing*, Vol. 16, 1989, pp. 73-82.
15. P.R. Gray and R.G. Meyer, *Analysis and Design of Analog Integrated Circuits*, 2nd ed., John Wiley & Sons, New York, 1984.
16. O. Rossetto, C. Jutten, and J. Herault, "Analog VLSI Synaptic Matrices as Building Blocks for Neural Networks," *IEEE Micro*, Vol. 9, No. 6, Dec. 1989, pp. 56-63.
17. K. Nagaraj, "Large-Swing CMOS Buffer Amplifier," *IEEE Trans. Solid-State Circuits*, Vol. 24, No. 1, 1989, pp. 181-183.



Hua Li is an assistant professor of computer science at Texas Tech University. His current research interests include computer vision, neural networks, and analog VLSI.

Li received the BS degree in electronic engineering from Tientsin University, China, and the MS and PhD degrees in electrical and computer engineering from the University of Iowa. He is a member of the IEEE, the Computer Society, and Upsilon Pi Epsilon.



Ching-Ho Chen is a graduate student in the Department of Computer Science at Texas Tech University. His main research interests include computer vision, analog VLSI, and neural networks. He received the BS degree in mathematics from the National Tsing Hua University in Taiwan and is a member of Upsilon Pi Epsilon.

Address questions about this article to Hua Li, Computer Science Department, College of Engineering, Texas Tech University, Lubbock, TX 79409; or e-mail at xdhua@ttacs1.ttu.edu.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 153

Medium 154

High 155

Signal analysis

continued from p. 15

Noninvasive conduction-velocity estimation. Muscle fiber conduction velocity is the propagation velocity of the depolarization along the membrane of muscle fibers. Conduction velocity is a basic physiological parameter related to

the type and diameter of the muscle fiber, to the pH of interstitial and intracellular fluids, to ion concentrations in different compartments, and to motor unit firing rate or stimulation frequency. (See the box for techniques used to estimate conduction velocity.)

Conduction velocity affects the power spectrum of the myoelectric signal,¹ and its variation contributes to spectral compression during the fatigue process.^{1,2}

Estimation of muscle-fiber conduction velocity

Among several other techniques, we describe those that are important for historical reasons or because of their clinical relevance.

Method based on spectral dips. The spectral-dips technique can be used with the bipolar detection technique (see the earlier box on signal detection). The myoelectric signal is filtered by the transfer function of the differential electrode. We can describe the squared modulus of such a transfer function with the following equation:¹

$$|H(j\omega)|^2 = K \sin^2 \frac{\omega d}{v} \quad (1)$$

where $|H(j\omega)|$ is the modulus of the transfer function of the differential electrode, K is a constant, ω is the radian frequency, d is the interelectrode distance, and v is the muscle-fiber conduction velocity.

Equation 1 shows that the transfer function of the modulus of the differential probe equals zero when $(\omega d/v)$ equals $2K\pi$. In other words, for frequencies satisfying Equation 2 below, the power spectral density function of the signal equals zero:

$$f = K \frac{v}{d} \quad (2)$$

where $K = \pm 1, \pm 2, \pm 3, \dots$

Consequently, we might estimate conduction velocity by measuring the frequency at which the first spectral dip occurs. In practice, however, this method is rarely applied. Under typical conditions, the first spectral dip cannot be detected because it occurs in a frequency band in which the power associated to the myoelectric signal is close to zero. Also, when the myoelectric signal generated during voluntary contractions is studied, the power spectrum estimate is affected by random errors that are generally difficult to reduce to a level allowing for reliable and accurate detection of spectral dips.

Cross-correlation. Another technique is to use the cross-correlation function to estimate the time delay between two myoelectric signals collected at two different sites along the active muscle fibers (see Figure D). If we assume the

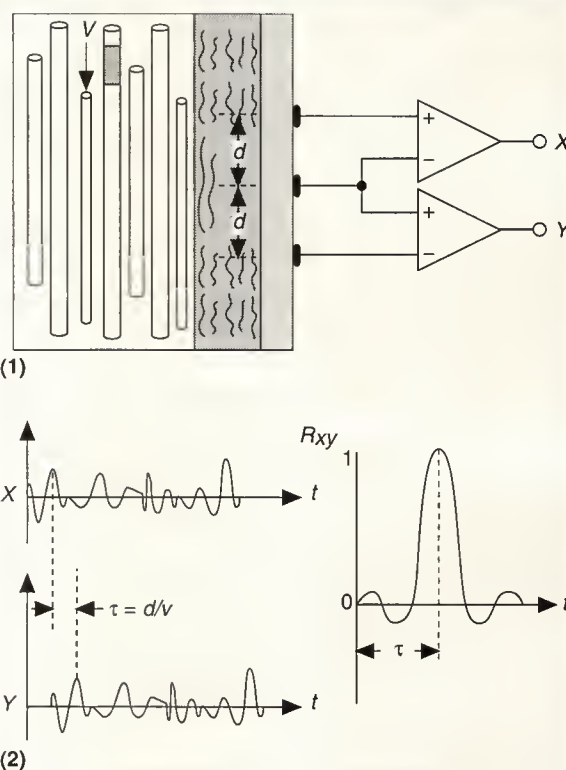


Figure D. Schematic representation of the cross-correlation technique: physiological model (1) and cross-correlation function between signals X and Y (2).

system is space-invariant within the detection area of the electrode, the two signals that result are time shifted by the interval $\Delta\tau$, as given by the following equation:

$$\Delta\tau = \frac{d}{v} \quad (3)$$

where $\Delta\tau$ is the time delay between the two signals, d is the interelectrode distance, and v is the muscle fiber conduction velocity.

Moreover, since conduction velocity relates to the size and histochemical properties of muscle fibers, observing it during different muscle activities may help in studying the recruitment strategy of motor units,³ and possibly in characterizing muscle architecture and fiber typing.^{2,3}

SMES processing techniques. During voluntary muscle contractions, the myoelectric signal contains information related to both the nervous system (the firing rate interactions of each motor unit) and the muscle tissue. When studying muscle fatigue, we need to separate the two contributions, especially the manner in which they influence the power-density spectrum of the myoelectric signal. Moreover, muscle fiber conduction velocity strongly influences the surface myoelectric signal power spectrum.^{1,4} But, it may also be affected by changes in the motor unit firing rate.

Electrical stimulation for muscle characterization. To characterize the muscle without any influence from the central nervous system, we can produce muscle contraction by using proper devices to simulate either nerve bundles or muscle motor points. A motor point is that area of skin above a muscle that shows the highest sensitivity to electrical stimuli applied with surface electrodes. During electrical stimulation, the firing rate of motor units can be kept constant, thus sepa-

rating the contribution of central and localized muscle fatigue. Figure 2 on the next page shows some examples of the time course of spectral parameters and conduction velocity during voluntary and electrically elicited contractions of the human *tibialis anterior*.

The observed time course of spectral parameters and conduction velocity obtained during electrically elicited and voluntary contractions shows that stimulated contractions yield smoother behavior of the parameters. A comprehensive comparison of the two techniques and a discussion of their potentialities is available elsewhere.^{2,3}

Myoelectric signal processing. Most information currently obtained from SMES analysis comes from the study of the time course of time and frequency parameters of the myoelectric signal generated during voluntary or electrically elicited contractions. Here we introduce some technical considerations in the computation of these parameters.

Earlier, we briefly presented the mathematical properties of SMES and pointed out that the myoelectric signal obtained during voluntary isometric contractions at constant force may be considered as a band-limited stochastic process with a Gaussian distribution of amplitudes.¹ In most cases, during relatively low-level and short contractions, the myoelectric

The cross-correlation technique relies on an important property of the cross-correlation function: If two signals are equal but time-delayed with respect to each other, the peak of their cross-correlation function has a value equal to one, and its localization on the abscissa equals the delay between the two signals. To estimate the conduction velocity of a group of muscle fibers, we need an electrode that collects two signals at two different sites along the muscle. The two signals are generally acquired by a computer, which estimates the cross-correlation function between them and calculates the abscissa of its maximum value. Given the relationship presented in Equation 3, if we know the interelectrode distance, we can obtain the value of the conduction velocity.

To obtain an acceptable resolution in the conduction-velocity estimation, we must compute the time delay between the two signals with a resolution of ± 10 to $20 \mu\text{s}$. Such a resolution can be obtained by either oversampling the signals or interpolating the cross-correlation function. Oversampling is the less efficient solution: It is burdensome computationally and requires the storage and manipulation of long sequences of data. Interpolation techniques increase the time resolution by interpolating the cross-correlation with a suitable function.

Spectral matching. The spectral matching technique is similar to the cross-correlation approach, but it is based on the minimization of the L_2 distance between the Fourier transforms of the delayed signals. Aligning two signals shifted in time by minimizing the mean-square difference between their amplitudes is equivalent to minimizing the mean-square error between their Fourier transforms.² Generally, to achieve an acceptable resolution, the alignment must be performed using fractions of the sampling period. This is easy in the frequency domain, because one signal can be time-shifted by any time interval τ simply by multiplying its Fourier transform by the complex number $e^{j\omega\tau}$.

Computationally, spectral matching is less expensive than the cross-correlation technique. CPU time may be reduced by a factor of 10.

References

1. L. Lindstrom and R.I. Magnusson, "Interpretations of the Myoelectric Power Spectra: A Model and Its Applications," *Proc. IEEE*, Vol. 65, 1977, pp. 653-662.
2. K. McGill and L. Dorfman, "High-Resolution Alignment of Sampled Waveforms," *IEEE Trans. Biomedical Eng.*, Vol. 31, No. 6, June 1984, pp. 462-468.

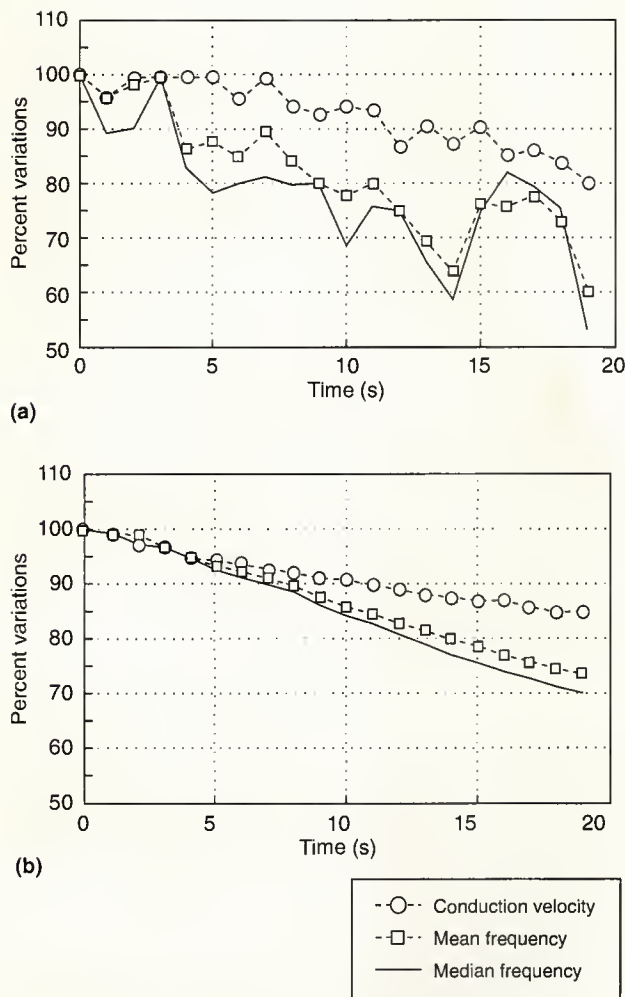


Figure 2. Examples of the time course of median frequency, mean frequency, and conduction velocity during contractions elicited voluntarily (80 percent of maximum voluntary contraction) (a) and electrically (b).

signal may be considered as *wide-sense stationary*. (A low-level contraction is 20 to 30 percent of the maximum voluntary contraction force, and a short contraction lasts for some tenths of seconds.) Evaluation of stationarity is of paramount importance when we compute the power spectrum of the voluntary myoelectric signal. When we increase the contraction level, the wide-sense stationary hypothesis generally holds during shorter epochs (0.5 to 1.5 seconds at 50 to 80 percent of the maximum voluntary contraction force).

Generally, we compute the time course of time and frequency domain parameters by subdividing the signal into epochs during which the wide-sense stationary hypothesis

holds. We then compute the values of the selected attributes over each epoch.

Different techniques are generally used to estimate the power spectrum density functions of voluntary or electrically elicited myoelectric signals. To process a voluntary myoelectric signal, the spectral estimation technique most frequently adopted relies on the estimation of the power spectrum density function, using the raw periodogram. After segmenting the signal into epochs in which the wide-sense stationary assumption holds, we compute the signal power spectrum simply by taking the squared modulus of the discrete Fourier transform of the signal itself. To increase computational efficiency, the discrete Fourier transform is generally computed with radix-2 fast Fourier transform algorithms.

In most clinical applications, the power spectrum is estimated principally to extract the time course of spectral variables—namely, the power spectrum mean and median frequencies. Although the raw periodogram is an inconsistent spectral estimator, results obtained in myoelectric signal processing are usually satisfactory thanks to the statistical properties of the mean and median frequencies.

During electrically elicited contractions, the signal is deterministic and quasiperiodic. The stimulation frequency imposed by the stimulator determines the period of the signal. (We call each electrical response of a muscle to a stimulus the M wave.) Since the signal is quasiperiodic, three different strategies are frequently used to compute its attributes:

- Subdivide the signal into epochs during which M waves may be considered as time-invariant. Then compute time and frequency attributes over each epoch.
- Compute amplitude attributes and the power spectrum over each single M wave. Since each single M wave is processed, the computational cost of this approach is generally high, and the time series of the computed attributes reflects random variations of the M waves. The spectral density of the signal may be easily interpolated by zero padding in the time domain to obtain the desired spectral resolution.
- Subdivide the signal into epochs during which M waves may be considered as time-invariant (as in the first method). Then compute the average M wave over each epoch and the amplitude and spectral parameters on the average M wave. If time averaging is performed properly, this approach combines the advantages of the other two methods.

Among the time domain parameters are the average rectified value, the root mean-square value, the muscle fiber conduction velocity, and the peak and peak-to-peak amplitudes, duration, and the area of the M wave. We compute these parameters after segmenting the input stream of samples as described earlier.

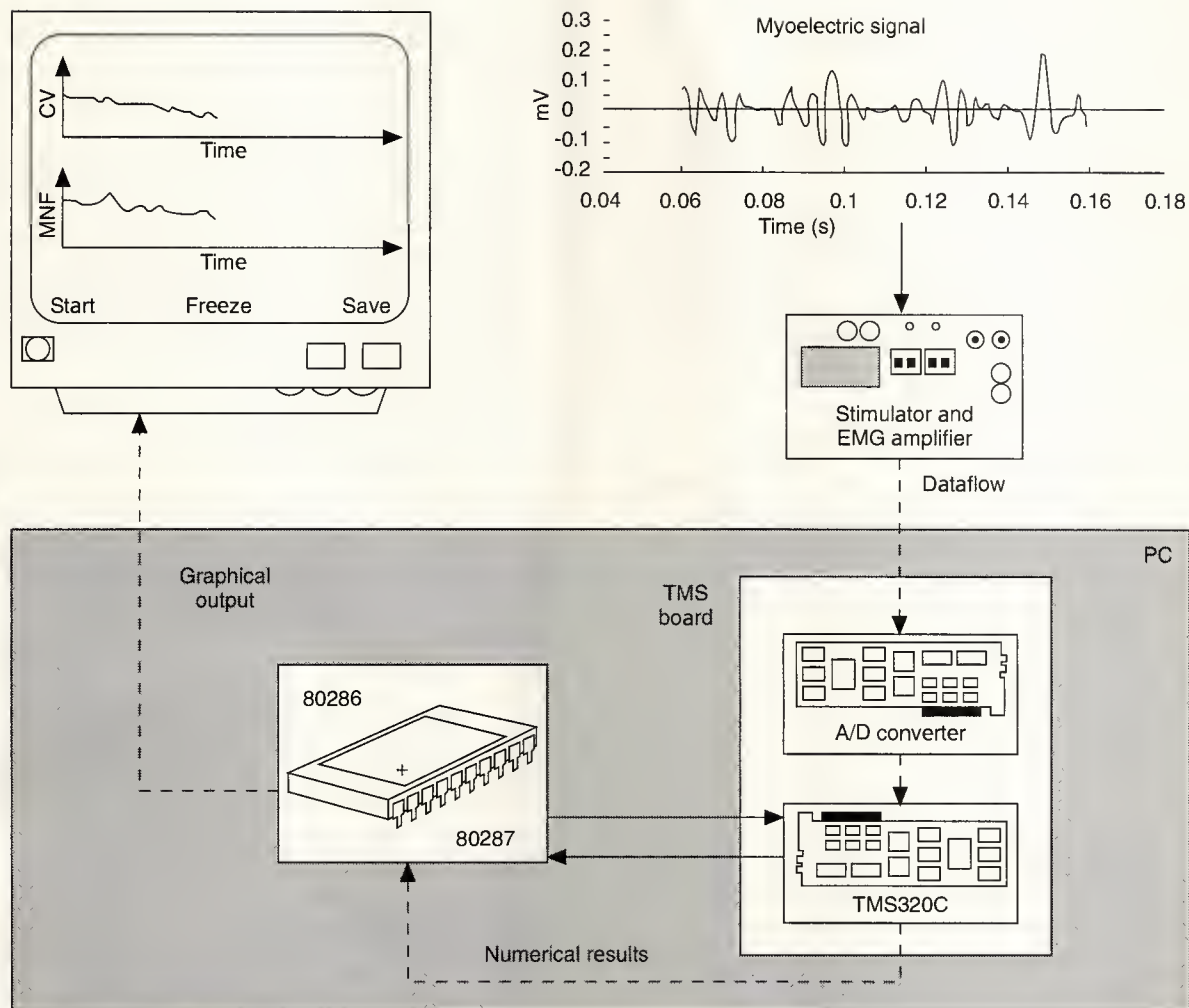


Figure 3. Schematic representation of a real-time myoelectric signal processing system. MNF indicates the mean frequency and CV the conduction velocity.

Real-time myoelectric signal processing. Off-line processing is well suited for research applications, but for most clinical applications on-line systems are better. In fact, if physicians can observe on line the time course of spectral and amplitude parameters during voluntary and electrically elicited contractions, they can promptly detect the occurrence of nonstationarities related to physiological events taking place inside the muscle. Examples are the onset of muscle fatigue or a change in the strategy used by the central nervous system to recruit motor units. These events may be used as feedback to allow the subject to control the contraction more effectively. Moreover, in applications of functional electrical stimulation for gait restoration, spectral and amplitude parameters of the myoelectric signal detected on line may be

used to control the stimulation strategy to increase muscle endurance as much as possible.

The availability of digital signal processing cards for industry-standard buses has made it possible to develop relatively low-cost, high-performance systems for on-line processing of biological signals. In this section, we describe a recently published system.

Figure 3 is a block diagram of a real-time myoelectric signal analyzer consisting of an 80286-based PC with a digital signal processing card inserted into a slot. The DSP card, from Loughborough Sound Images, Ltd., is based on the Texas Instruments TMS320C25 DSP chip. Its principal features are a clock frequency of 40 MHz, a 64-Kbyte dual-port memory for both data and code, and an on-board 16-bit analog-to-digital

converter. The host computer and the DSP processor work in parallel.

The host computer

- manages the user interface so the user can store the patient's general data and supervise the display of the results,
- initializes the processing cycle and downloads the TMS320 executable code from the host computer's hard disk to the program memory of the DSP board, and
- manages the video operations through direct addressing of the EGA or VGA card memory to obtain the real-time display of the results.

At the same time, the DSP subsystem

- manages the interrupts generated by the end of conversions coming from the analog-to-digital converter,
- computes the 128-point fast Fourier transforms and evaluates the short-term power spectrum,
- evaluates maximum, mean, and median frequencies of the power spectrum,
- smooths the spectral parameters using a four-point moving average,
- pseudocolor-codes the spectral power densities, and
- transfers the results to the host computer for real-time display.

In a typical processing cycle, the system is initialized with user-supplied parameters, and the host computer downloads the desired DSP routine from its hard disk to the DSP card's program memory. The host computer starts the process by resetting the TMS320C25 and then waits for the first frame of results. When the first frame is available, the two processors start working in parallel. The DSP subsystem performs two different tasks:

- It services the interrupts generated by the analog-to-digital converter during the current subepoch $t(j)$ when converted data are available, storing them in the data memory;
- It processes data stored in the subepoch $t(j-1)$.

At the same time, the host computer displays the results obtained by processing the data acquired in the subepoch $t(j-2)$.

The delay of two subepochs (256 ms for a sampling frequency equal to 1 kHz) between data acquisition and graphic display of the results is acceptable for clinical applications. Although the throughput of the system is slightly higher than 20 kHz, the user may select lower sampling rates to avoid overloading the system. Sampling rates ranging from 1 kHz to 2 kHz are generally satisfactory. A synchronization bit

from a TMS320C25 register synchronizes the two processors.

While the system collects and processes data in the background, the menu-driven user interface runs in the foreground. The user chooses among several options with function keys:

- Stopping or starting an evaluation session.
- Displaying in real time the color-coded power spectrum density functions and their mean and median frequencies. A freeze option lets the user stop the video-display update without stopping data acquisition and processing, and a record option saves the values of the frozen screen into an ASCII file for future use.
- Varying the sampling rate.
- Evaluating and displaying in real time the confidence intervals for spectral parameters.
- Evaluating power spectrum density functions and spectral parameters averaged over subsequent subepochs, and displaying the results in three different graphics windows.
- Displaying the myoelectric signal in the time domain to verify the signal quality on the host computer's video monitor.

This system is powerful enough to compute muscle fiber conduction velocity on line using the spectral matching technique described in the earlier box on conduction-velocity estimation. It provides sampling frequencies up to 5 kHz and collects data from up to four different channels. The adjacent box on clinical applications describes some uses of such systems.

Needle electromyography

Needle electromyography techniques obtain diagnostic and/or prognostic data by analyzing the myoelectric signal. Although most data are obtained from visual inspection of the interference pattern, in the last decade several techniques that use computers have been introduced into clinical practice. Physicians have shown appreciable interest in the decomposition of the interference pattern and macro EMG techniques. Several myoelectric signal analysis systems now include them as options.

We briefly describe these two techniques to show typical requirements for a computer system used in this field. The literature provides more detailed descriptions of the decomposition problem¹ and the macro EMG technique.⁵

Decomposition of the interference pattern. The myoelectric signal is the sum of the action potentials produced by each particular active motor unit. The decomposition of the interference pattern procedure decomposes the myoelectric signal into its constituent MUAP trains.¹ Researchers and physicians use this technique to investigate the strategies implemented by the neuromuscular system for controlling muscle contractions.

Clinical applications of surface electromyography

Two clinical uses of SMES analysis are muscle fatigue evaluation and noninvasive fiber typing.

Muscle fatigue evaluation. SMES analysis has been used extensively in the study of muscle fatigue. Analysis and quantification of muscle fatigue are crucial in sport, occupational, and rehabilitation medicine. Fatigue is a continuous process that begins at the onset of a muscle contraction. It has two components: fatigue of the nervous system and localized muscle fatigue. The former is represented by the variation of the discharge activity of motor units during a sustained muscle contraction and is related to processes taking place in the nervous system. The latter is related only to processes taking place inside muscles. A comprehensive review of muscle fatigue is available elsewhere.¹

Since the beginning of this century, the frequency components of the surface myoelectric signal have been known to decrease during sustained contractions. Nonetheless, physiologists have become accustomed to describing (and measuring) muscle fatigue by monitoring the force output of the muscle. They usually refer to the onset of fatigue as a point in time when the subject cannot sustain the force output at a preset value, and measure fatigue as the continual decrease of force production.

This approach has drawbacks. It works best when the muscle activity is controlled artificially, for example, by electrical stimulation of the nerve or muscle under study. During voluntary contractions, the sincerity of the subject's

effort influences the level of force output. Also, during less-than-maximum-force contractions, a muscle can sustain a constant-force output for a definite amount of time before a decrease in force occurs. Nevertheless, during this interval the force-generation characteristics of the muscle fibers (mechanical properties and control properties of the motor units) change as a function of time. These modifications, which have a strong bearing on the behavior of the muscle, are excluded from the evaluation of muscle fatigue.

The contractile measure of muscle fatigue is not always useful from a pragmatic point of view. For example, if we want to monitor the rate of muscle fatigue during meaningful and productive muscle efforts, a measure of contractile fatigue would be useless because it would only indicate the onset of exhaustion experienced by the subject.

On the other hand, spectral parameters allow the clinician and researcher to assess the evolution of the fatigue process from the beginning of the contraction, long before mechanical manifestations of muscle fatigue are evident. Figure E illustrates this concept. It shows the time course of the mean frequency of the SMES power spectrum and of the force output of a muscle during a maximal voluntary contraction. The power spectrum mean frequency starts decreasing from the beginning of the contraction, while the muscle fails to produce the desired effort 13 seconds later.

The myoelectric signal power spectrum is affected by the firing rate of motor units, by the cancellation effect occurring because of the random superposition of motor unit action potentials, and by the shapes of the action potentials. During sustained contractions, all these factors change, but there is evidence^{1,2} that the power-spectral compression toward lower frequencies results mostly from widening of the shapes of the motor unit action potentials. By studying the relationships among variations of spectral and amplitude parameters and conduction velocity, we can obtain much information about processes taking place inside the muscle tissue.

Recently, several dedicated devices have been designed for assessing changes of the power spectrum of the surface myoelectric signal due to localized muscle fatigue. Among others, the muscle fatigue monitor³ has been used extensively for studying muscle fatigue associated with lower back pain, and a similar device based on a digital signal processing board has been described by Balestra et al.⁴ In such applications, computers are necessary to realize devices suitable for clinical use.

Noninvasive fiber typing. We can divide muscle fibers into several families that differ in their metabolism,

continued on p. 54

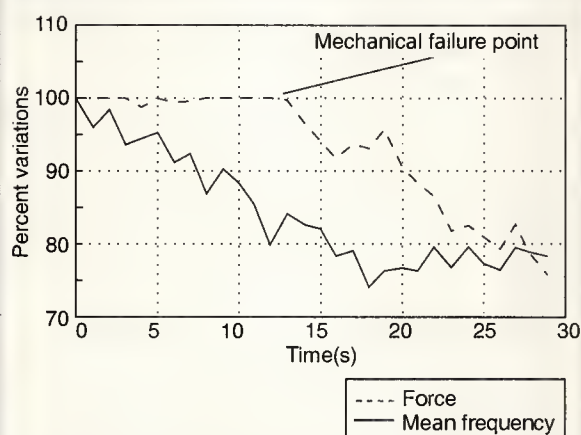


Figure E. Time course of muscle force and mean frequency of the SMES power density function during an attempted constant-force contraction.

continued from p. 53

mechanical properties, and cross-section and electrophysiological characteristics. The distribution of different muscle fibers inside a skeletal muscle is related to the particular muscle, its activity, and the subject's training protocols and age. Neuromuscular diseases can cause a variation of the population of fibers inside a muscle, thus changing its properties and characteristics.

Data about muscle fiber constituency help a clinician trace the progression of neuromuscular diseases or the effectiveness of a training protocol. Although precise information about fiber distribution inside a muscle is obtained only with a biopsy, qualitative data can be obtained by analyzing the time course of SMES parameters and conduction velocity during fatiguing contractions. In fact, spectral parameters and muscle fiber conduction velocity relate to the size and type of muscle fibers.

By analyzing the surface myoelectric signal, a clinician can infer the recruitment order of motor units during voluntary and electrically elicited contractions. According to Henneman's size principle, during voluntary contractions of increasing force, larger motor units (made up of larger muscle fibers) are progressively recruited.

The analysis of the motor unit recruitment order is particularly important when studying pathologies known to affect it substantially. Also, for applications using functional electrical stimulation (for example, gait restoration in paraplegic patients), the recruitment order is important because it affects the endurance of electrically stimulated muscles.

References

1. J. Basmajian and C.J. De Luca, *Muscles Alive*, 5th ed., Williams and Wilkins, Baltimore, 1985.
2. R. Merletti, M. Knaflitz, and C.J. De Luca, "Myoelectric Manifestations of Fatigue in Voluntary and Electrically Elicited Contractions," *J. Applied Physiology*, Vol. 69, No. 5, 1990, p. 1810-1820.
3. L.D. Gilmore and C.J. De Luca, "Muscle Fatigue Monitor (MFM): Second Generation," *IEEE Trans. Biomedical Eng.*, Vol. 32, No. 1, Jan., 1985, pp. 75-78.
4. G. Balestra et al., "On-Line Estimation of Surface Myoelectric Signal Spectral Parameters: Methodological Considerations," in *Electrophysiological Kinesiology*, P.A. Anderson, D.J. Hobart, and J. Danoff, eds., Elsevier, Amsterdam, 1991, pp. 11-14.

Different algorithms to obtain a reliable decomposition procedure have been proposed in literature.⁶⁻⁸ For clinical purposes, physicians most often consider the MUAP morphological characteristics, while researchers examine both the MUAP shape and the motor unit's firing history to investigate motor unit properties and the strategies implemented by the central nervous system for motor control.

Figure 4 shows how a typical decomposition algorithm works. The input is the interference pattern collected by a needle electrode inserted into the muscle to be studied. Such input appears as a *noiselike* process. Here, the procedure must recognize the contribution of single motor units, classify the different MUAP shapes, and extract the firing history of each active motor unit. The output of the algorithm is a database with the shapes of all the MUAPs detected by the system and a structure of their firing histories.

The accuracy of a decomposition algorithm used in research must be very high. In 1973, Shiavi and Negin demonstrated that a misclassification of one or two MUAPs out of 100 may compromise the possibility of observing important characteristics of the motor unit behavior.¹

Statistical analysis of the motor unit firing histories typically requires the acquisition and processing of signal records ranging from 5 to 60 seconds long. Moreover, to obtain an acceptable reconstruction of the MUAP shape in the time domain, systems oversample the signal at sampling frequencies as high as 50 kHz. To improve recognition and classification algorithm performance, several channels are acquired simultaneously, thus increasing memory use and processing time considerably.

From a technical point of view, the essential problems are data compression and computational speed. Besides, current decomposition algorithms generally require a large amount of man-machine interaction, so graphics must be advanced to facilitate and speed the operator's work as much as possible. Until a few years ago, only minicomputers with large mass memory were suitable for processing these algorithms. Now, powerful PCs and workstations support most decomposition programs, and accurate and reliable systems are available and affordable. Thus, in the near future, EMG decomposition will probably become a widely used technique.

The following subsections outline the steps generally involved in a decomposition technique.

Signal acquisition. Usually the system acquires data from more than one channel, and the sampling rate may range from 10 kHz to 50 kHz. The duration of the acquisition may range from a few seconds to a minute. During acquisition, the computer screen displays the signals to allow the operator to control the quality of the collected signals. This task is particularly important because of the small detection volume of the needle electrodes that collect the signals. Even a small movement of the needle may totally change the pool of motor

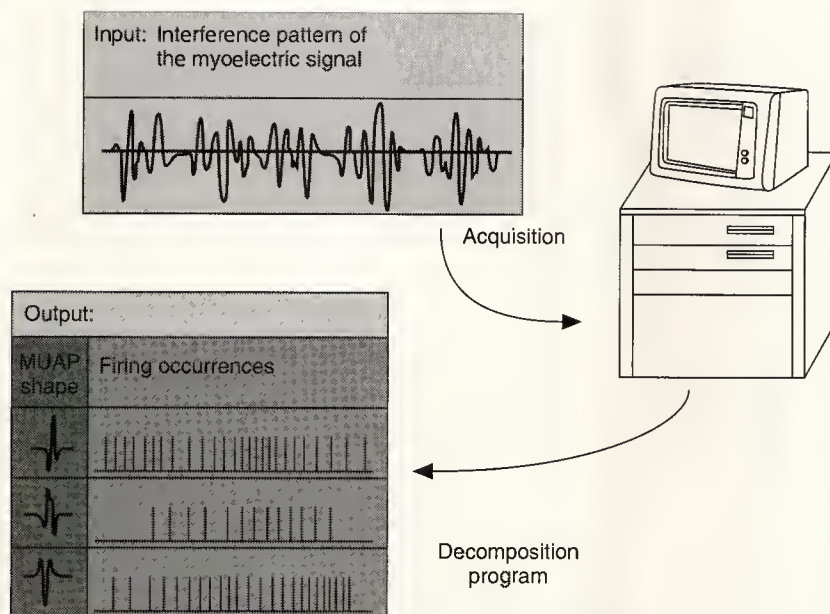


Figure 4. A sketch of the decomposition process. The waveforms reported in the output section represent the MUAP shape of three different motor units that were identified by the process. The diagram on the right side is a time diagram of the firing history of the corresponding motor unit. Each bar corresponds to a firing occurrence.

units observed, thus compromising the possibility of assessing the firing behavior of a certain pool of motor units over a sufficiently long time interval.

The acquisition system consists of an IBM PC AT-class machine or a workstation with a suitable analog-to-digital converter board and a 19- or 20-inch high-resolution screen. The tasks in this phase are data collection and simultaneous display. A data-compression algorithm is usually necessary to reduce signal redundancy and avoid the storage of too much data. Intelligent analog-to-digital converter cards with their own microprocessor system and possibly a DSP chip can greatly facilitate the on-line processing and display of the signals.

Identification of individual MUAPs. The identification of individual MUAPs often requires an improvement of the signal-to-noise ratio. Powerful techniques for noise reduction are currently available—for example, optimal Wiener filtering, adaptive filtering, and noise reduction through the use of high-order spectra. But identification remains a challenging task, especially if the initial signal-to-noise ratio is relatively low. For data compression to optimize memory occupation, a threshold value is set either automatically or manually, taking into account the statistical properties of the recorded signal. The acquired samples are compared with

that threshold. Only values over the threshold are saved, so memory isn't wasted to record noise.

Resolution of superpositions. The summation of different MUAPs belonging to motor units that fire simultaneously leads to a waveform in which distinguishing each single MUAP is generally difficult. Such a waveform is called a *superposition*. At low contraction levels (below 20 to 30 percent of the maximum voluntary contraction force) the MUAP shapes are usually well separated, but the likelihood of observing superpositions rises with increasing contraction levels. At contraction levels higher than 60 to 80 percent of the maximum voluntary contraction force, most MUAP shapes are superimposed, making it difficult to identify the active motor units.

The algorithms used to separate the complex waveforms into their constituents are very time-consuming. In fact, when distinguishing any previously classified MUAP shape is not possible and the record under observation cannot be classified as a new MUAP shape, superpositions are typically resolved by computing the probability that every motor unit previously classified fires at a given instant. After finding the motor unit that maximizes the probability of firing at that time, the program subtracts the corresponding MUAP shape from the record. The same algorithm is recursively applied to the remaining signal until no more MUAPs can be identified. The program evaluates the result according to a set of rules, and, if it is not satisfactory, starts the process again and alerts the human operator to control its progression.

MUAP classification. The classification techniques used by most decomposition algorithms are based on pattern matching. At the beginning of the analysis of a new signal record, both the number of motor units whose firing can be detected and their characteristics are unknown. When a new motor unit is identified, its shape is recorded and considered as a new template. This template is then added to the template database and used to classify other MUAPs belonging to the same motor unit. Every time the program recognizes a waveform as generated by a specific motor unit, it updates the corresponding template to keep track of the MUAP modifications caused by fatigue, small electrode movements, or other physiological variations. Artificial intelligence techniques can improve both the accuracy and speed of the classification process.

Measuring temporal and morphological information. The outcomes of the decomposition process are the characteristics (MUAP shape, firing history, and so on) of each detected motor unit. At the end of the process the researchers evaluate these features to extract the physiological information.

If the purpose is to better understand the control properties of motor units, researchers are interested mostly in their firing histories. Firing history is usually represented as a sequence of δ functions placed along the time axis. Figure 4 shows an example of such a plot. The distance between two subsequent firing occurrences is called the interpulse interval. Counting the number of firing occurrences of a particular motor unit within a given period of time and dividing that number by the period of time itself yields the so-called mean firing rate, which correlates with physiological and anatomical characteristics of the muscle fibers that constitute that motor unit.

Typically, physicians can extract information about the physiological condition of muscle fibers belonging to a given motor unit by observing variables related to the MUAP shape. Figure 5 summarizes the main variables related to the MUAP shape, which is generally computed by averaging several MUAPs belonging to the same motor unit to reduce pulse-to-pulse variability.

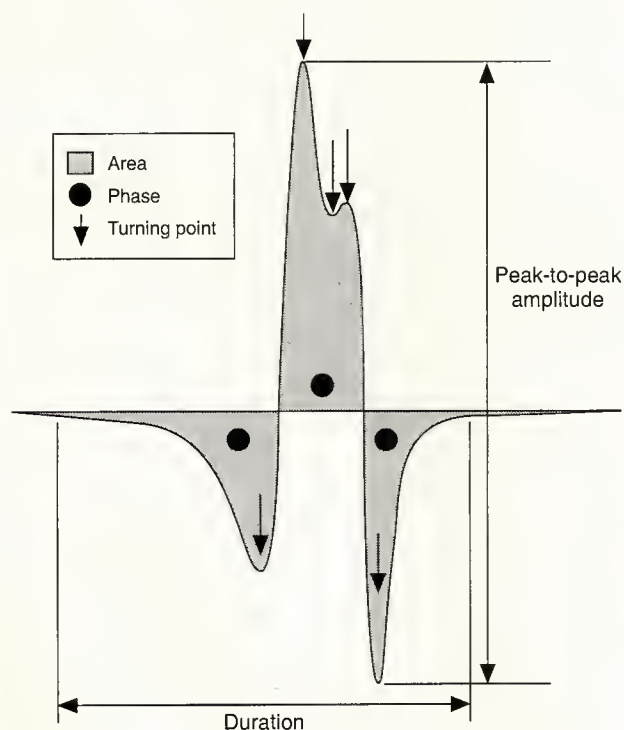


Figure 5. MUAP shape characteristics.

Decomposition using artificial intelligence. In 1988, Broman proposed a decomposition program based on AI techniques.⁹ The program consists of three planes: control, module, and data. The data plane contains the raw and processed data, as well as data to control the selection of the computing modules. The module plane contains the basic processing modules that deal with raw data, and the refinement modules applied to processed data. The module plane also contains an interface between the data plane and the model in the control plane. The control plane contains

- a priori knowledge about the physiology of the myoelectric signal and possible artifacts of the recording technique,
- a scheduler that selects the appropriate module for the actual processing, and
- a dynamic model with information about the decomposition process under development.

This approach significantly increases the computational speed and dramatically reduces the need for human interaction (at least with signals without numerous and complicated superpositions). Nevertheless, it is still far from being adequate to extensive clinical application.

Macro EMG. Stalberg⁵ described the macro EMG technique in 1980 to extract potentials belonging to a single motor unit for motor unit classification and study. General-purpose needle electrodes have a detection volume large enough to collect MUAPs generated by muscle fibers belonging to a number of different motor units. In pathological conditions, normal and pathological motor units can coexist in the same muscle. Thus, with general-purpose needles it is generally difficult to obtain information about either motor unit type.

The macro EMG approach involves an operation that averages a given number of frames of myoelectric signals collected by the general-purpose needle electrode. The single-fiber action potential generated by a fiber belonging to the motor unit to be studied triggers the averaging process. We adjust the position of the single-fiber needle to select a stable single-fiber potential to trigger an averager. The macro electrode collects signal epochs of some tens of milliseconds (typically 60 ms) after each triggering event. These are fed to the averager, which returns the macro potential corresponding to the fiber potential trigger. If we assume the firing histories of the motor units detected by the large needle electrode are uncorrelated, and if the number of averaged frames is large enough, the averaging process should lessen potentials generated by all the motor units except those synchronized with the trigger.

By changing the position of the single-fiber needle in the muscle, we can trigger the averager with muscle fiber potentials belonging to different motor units. This lets us extract consecutively from the background noise the macro MUAPs

***With PCs and workstations,
clinicians can apply the signal
processing techniques previously
developed in the field of basic
research.***

that occur time-locked to each fiber spike train used as a trigger source. Thus, we can determine the number of muscle fibers that constitute the motor unit, and obtain information about the temporal and spatial scattering of muscle fiber depolarization inside each motor unit. This technique is very useful because it offers information when other techniques give insufficient or misleading results.⁵

Systems designed to support macro EMG analysis resemble those used for myoelectric signal decomposition. They acquire two signals of interest (concentric and the macro EMG) at a sampling rate ranging from 10 kHz to 50 kHz. Epochs as long as 100 to 120 seconds are generally studied.¹⁰

Expert systems

In the last 10 years, expert systems to help the physician diagnose neuromuscular disorders have been developed. From those described in the literature, we consider some systems to be typical:

- The Localize system localizes the site of a lesion within peripheral nerves using muscle-strength testing (1982).¹¹
- Myosis diagnoses mono- and polyneuropathies using the myoelectric signal (1985).¹²
- Blinowska and Verroust's system helps the physician diagnose the carpal tunnel syndrome (1987).¹³
- A system described by Gallardo et al. helps in the study of plexus and root lesions (1987).¹⁴
- Kandid assists in the diagnosis of the complete range of neuromuscular disorders and supports the physician in planning an optimal sequence for testing.¹⁵

The most complex expert system in the electromyographic field today is Munin. This knowledge-based system supports an electromyographer in the various steps of an EMG examination performed with needle electromyographic techniques.¹⁶

The reasoning performed by Munin is based on hypothesis-and-test methodology. A typical consultation starts with the physician entering the patient's anamnestic data. Using this information, the system generates a ranked list of EMG

tests, ordered according to their expected diagnostic discrimination. The electromyographer selects a test, and Munin helps carry it out.

Munin uses the results to perform a two-step reasoning process. The first step describes the pathophysiological changes of the muscles and nerves involved in the test, while the second lists the diseases that may cause the changes recognized by the first step. To each disease the system assigns a probability score. At this point, the physician may either select a new test or accept the diagnostic results. If a new test is performed, Munin updates the list of the possible diseases and their probabilities. The consultation stops when a diagnosis is reached, when no more tests are available, or when the disadvantages of performing new tests are greater than the expected diagnostic benefits.

MODERN EMG SIGNAL ANALYSIS STARTED in the early 1960s, when the first computer systems became available to researchers. Only recently has the availability of microprocessor-based personal computers and workstations allowed clinicians to apply the signal processing techniques previously developed in the field of basic research. Here, we reviewed the most recent applications of computers to the analysis of the EMG signal, specifically describing several techniques developed in the last two decades. We tried to focus our attention on the procedures that are presently part of standard clinical practice; however, the development of new techniques is ongoing and the field of applied research continuously expanding.

We are currently able to extract only a small amount of the information on the neuromuscular system that the EMG signal embodies. The challenge for the future is to increase as much as possible our ability to understand the neuromuscular system by continuing to analyze the electrical activity of muscles and nerves. **□**

Acknowledgments

We wish to acknowledge C.J. De Luca's and R. Merletti's indirect contributions to this work. They introduced us to the field of computer analysis of the myoelectric signal, and we extensively discussed with them most of the issues dealt with in this article.

References

1. J. Basmajian and C.J. De Luca. *Muscles Alive*, 5th ed., Williams and Wilkins, Baltimore, 1985.
2. R. Merletti, M. Knaflitz, and C.J. De Luca, "Myoelectric Manifestations

- of Fatigue in Voluntary and Electrically Elicited Contractions," *J. Applied Physiology*, Vol. 69, No. 5, 1990, p. 1810-1820.
3. M. Knaflitz, R. Merletti, and C.J. De Luca, "Inference of Motor Unit Recruitment Order in Voluntary and Electrically Elicited Contractions," *J. Applied Physiology*, Vol. 68, No. 4, 1990, p. 1657-1667.
 4. L. Lindstrom and R.I. Magnusson, "Interpretations of the Myoelectric Power Spectra: A Model and Its Applications," *Proc. IEEE*, Vol. 65, 1977, pp. 653-662.
 5. E. Stalberg, "Macro EMG, A New Recording Technique," *J. Neurol. Neurosurg. Psychiatry*, Vol. 43, 1980, pp. 475-483.
 6. R.S. LeFever, A.P. Xenakis, and C.J. De Luca, "A Procedure for Decomposing the Myoelectric Signal into Its Constituent Action Potential," *IEEE Trans. Biomedical Eng.*, Vol. 29, No. 3, Mar. 1982, pp. 149-164.
 7. S. Andreassen, "Computerized Analysis of Motor Unit Firing," in *Computer-Aided Electromyography*, "Progresses in Clinical Neurophysiology," Vol. 10, J.E. Desmedt, ed., Karger, Basel, Switzerland, 1983, pp. 150-163.
 8. L.J. Dorfman and K.C. McGill, "Automatic Quantitative EMG," *Muscle Nerve*, Vol. 11, 1988, pp. 804-827.
 9. H. Broman, "Knowledge-Based Signal Processing in the Decomposition of Myoelectric Signals," *IEEE Eng. in Medicine and Biology*, Vol. 7, No. 2, June 1988, pp. 24-28.
 10. P. Guiheneuc, C. Doncarli, and M. Le Bastard, "Concomitant Multitrain Automatic Analysis of Motor Unit Potentials and Macro EMG," in *Computer-Aided Electromyography and Expert Systems*, J.E. Desmedt, ed., Elsevier, Amsterdam, 1989, pp. 241-247.
 11. M.B. First et al., "Localize: Computer-Assisted Localization of Peripheral Nervous System Lesions," *Comput. Biomed. Res.*, Vol. 15, 1982, pp. 525-543.
 12. A. Vila, D. Ziebelin, and F. Reymond, "Experimental EMG Expert System as an Aid in Diagnosis," *Electroenceph. Clin. Neurophysiol.*, Vol. 61, No. S240, 1985.
 13. A. Blinowska and J. Verroust, "Building an Expert System in Electrodiagnosis of Neuromuscular Diseases," *Electroenceph. Clin. Neurophysiol.*, Vol. 66, No. S10, 1987.
 14. R. Gallardo et al., "Artificial Intelligence in EMG Diagnosis of Cervical Roots and Brachial Plexus Lesions," *Electroenceph. Clin. Neurophysiol.*, Vol. 66, No. S37, 1987.
 15. A. Fuglsang-Frederiksen and S.M. Jeppesen, "A Rule-Based EMG Expert System for Diagnosing Neuromuscular Disorders," in *Computer-Aided Electromyography and Expert Systems*, J.E. Desmedt, ed., Elsevier, Amsterdam, 1989, pp. 289-296.
 16. S. Andreassen et al., "MUNIN—An Expert EMG Assistant," in *Computer-Aided Electromyography and Expert Systems*, J.E. Desmedt, ed., Elsevier, Amsterdam, 1989, pp. 241-247.



Marco Knaflitz is a research associate in the Department of Electronics at the Politecnico di Torino, Italy. His principal research interests include the design of instruments, algorithms, and software for surface myoelectric signal processing. In the last five years he spent 18 months working in this area at the Neuromuscular Research Center of Boston University.

Knaflitz received the Italian Laurea in electrical engineering and a PhD in electronics from the Politecnico di Torino. He is member of AIIMB (Italian Association of Biomedical Engineers) and the International Society for Electrophysiological Kinesiology.



Gabriella Balestra holds a postdoctoral grant at the same department. Her research interests include the development of processing techniques and algorithms for biological signals, and particularly the application of AI techniques to biomedical signal processing. In 1989 and 1990 she spent six months as a visiting scientist at the Neuromuscular Research Center of Boston University, where she worked on computer simulation of the myoelectric signal.

Balestra received the Italian Laurea in computer science from the University of Turin and a PhD in Systems and Informatics from the Politecnico di Torino. She is a member of AIIMB and the International Society for Electrophysiological Kinesiology.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

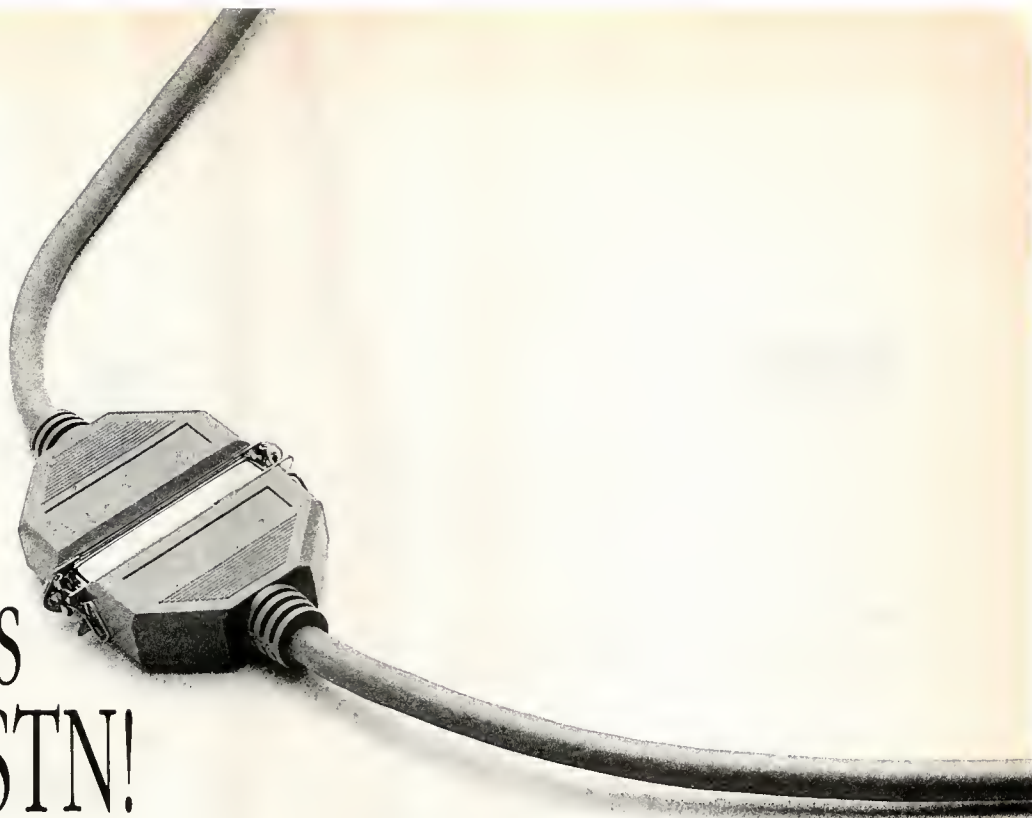
Low 156

Medium 157

High 158

Address questions about this article to Marco Knaflitz, Dipartimento di Elettronica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy.

Make Great Connections Online On STN!



On STN International®, you can access databases covering every area of engineering. You'll find bibliographic and numeric files produced by leading scientific organizations, like AIChE, Engineering Information, IEEE, National Institute of Standards and Technology, and many others.

And everything about STN has been created to assist you in obtaining this information efficiently and economically. On STN, you'll find special features which enhance your searching, whether you're a novice or an expert.

One Command Language —
Using a few simple commands, you'll be able to obtain information in more

than 100 databases on STN. And STN's software, Messenger®, enables you to carry a search created in one database over to another on STN for further information.

Element Term Index —

Through STN's Element Term (ET) Field, you can search for chemical symbols and other specialized notations. Using the ET Field can increase your accuracy AND efficiency.

Numeric Searching —

On STN, you can search numeric values to locate substances having the property values you specify; search numeric ranges to find data you might otherwise miss; choose from SI, metric, engineering, or other units to

display property values; search with substance names, names of properties, or CAS Registry Numbers® to retrieve numeric data.

As an STN customer, you can receive help from workshops, tutorial diskettes, STN Express® software, toll-free Help Desk, newsletters, and online document ordering. No one supports you like STN!

Make great connections on STN by filling out and returning the coupon below.

☐ **YES! Please tell me how to become a user of STN International.**

Name _____

Title _____

Organization _____

Address _____

City, State ZIP _____

Phone _____

MAIL TO: STN International, c/o Chemical Abstracts Service, Marketing Dept. 30091,
P.O. Box 3012, Columbus, OH 43210
FAX TO: 1-614-447-3713



Alphorn

continued from p. 19

must match the exported definitions of the server module. Figure 4 shows a client that calls a server procedure when both modules are directly linked. The shaded key form symbolizes the interface.

Remote procedure calls

A remote procedure call looks exactly like a local procedure call to another module of the same task. For instance:

```
Bank.GetItem ("SMITH", Salary);
```

This call is issued by a client program to a server procedure `Bank.GetItem`. The client is suspended until the call returns. That is exactly how the familiar local procedure call works. Figure 5 shows a remote procedure call as a function of time.

In a local call, the called procedure belongs to another module of the same task. Communication takes place directly through procedure call mechanisms the compiler provides. This is the view of communication the programmer ideally should have.

In a remote call, the server belongs to a different task of the same or a different processor. For the client, however, everything should happen as if the call were local. In reality, the call is forwarded over the communication path. `Bank.GetItem` is executed by another task in the same computer (internal call) or in another computer (external call), which can be a different type of machine or can be running a different operating system. The results are returned in another message, as shown in Figure 6.

Remote procedure calls are very similar to local procedure calls. For instance, remote calls may be nested at will. But there are some differences:

- **Local variables.** Although most languages permit a module to export local variables, this is not allowed if the module is to be distributed; client and server residing on different sites share no common address space. Therefore, no reference to the address space of either the client or the server (common variables) may be used. This rule is in accordance with object-oriented programming. All information is transmitted in the parameters of the called procedure. In remote calls, parameters are always transmitted by value (even if the programmer passes parameters by reference).
- **Parallelism.** In contrast to a local procedure call, the same service may be called again by another client while a remote procedure call is executing. For each client, a different instance of the service is started. Each service is executed by a different thread. When the threads are

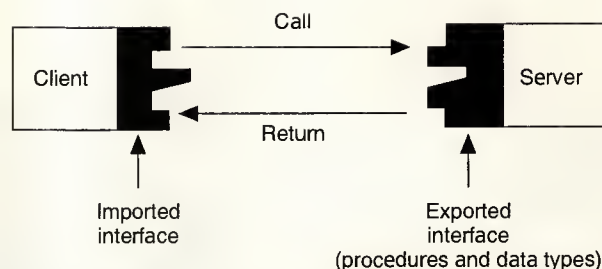


Figure 4. Interface of a called procedure.

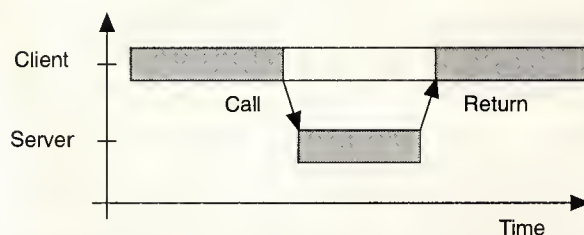


Figure 5. Remote procedure call.

exhausted, the calls are queued. The threads need synchronization among themselves to access common resources.

- **Blocking calls.** Although client and server may be executed by independent processors, they do not execute in parallel; the client remains blocked until the call returns, according to the semantics of a procedure call. A client may wait for the result of only one call at a time. This causes no loss of parallelism, however. The processor that was executing a blocked client is free to execute another service, the same service again (on behalf of another client), or another client until the call returns. When a call is blocked, another activity can start or resume execution.
- **Nonblocking calls.** In addition to blocking calls, a client may start multiple calls, provided these calls do not return results. Such calls are called remote procedure invocations. Unlike RPCs, RPIs require an underlying flow control mechanism, invisible to the user. A client may not wait for the completion of an RPI. Unlike Delta-4, Alphorn supports no asynchronous call with results (Request Service Request/Remote Service Wait), because this construct loopholes the structure. (In ANSA, blocking and nonblocking calls are called synchronous and asynchronous.)
- **Events.** Normally, a remote call is a one-to-one relation. Alphorn has no broadcast calls but can publish an event from a signaler to subscribed handlers. Programmers insisted on having this construct, but it does not support

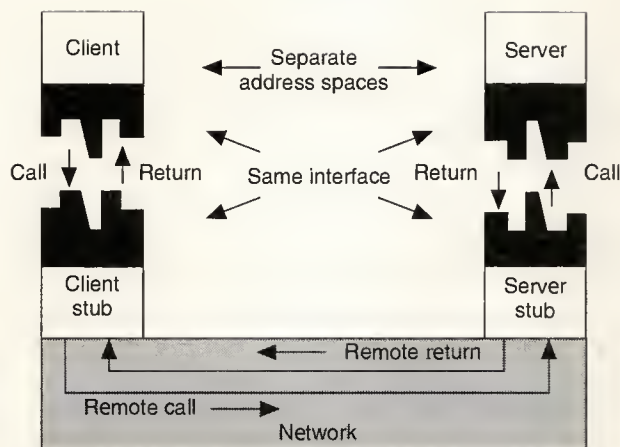


Figure 6. Client and server stubs.

fault tolerance and we therefore do not encourage its use.

Stubs

The entity in charge of relaying the call is called a stub (see Figure 6). The client stub plays the role of a server for the client. The client accesses the client stub in the same way it would access a local service procedure. The client stub converts the call into a message, which is sent over the network or over shared memory to the destination site. There, it is received by the server stub, which plays the role of a client for the server on that site. The server executes the service and returns to the server stub, which forwards the result to

the client stub. The client stub then returns the results in the same way a local service procedure would do.

Some languages, such as Argus,⁹ provide remote procedure calls as programming constructs. Most other languages cannot express that a procedure is called remotely or declare a module as server. Nor is this necessary, since we want to handle local calls and remote calls identically. To preserve the same interface for remote procedure calls as for local calls, we use stubs, not only for communication, but also as interfacing tools. The client stub presents the same interface to the client as the actual server would. In fact, the client stub simulates the existence of a local server. The server stub simulates a local client call to the actual server. The call interface is the same as it would be if the server were called locally. This is illustrated in Figure 6.

Stubs do much more than just transform a local call into a remote call. In fact, they are the centerpiece of the communication process. Stubs initialize and administer communication, control the success of calls, publish services over the network, and provide the hooks for debugging tools. In fault-tolerant systems, stubs control the correct replication and synchronization of redundant processors.¹⁰ In heterogeneous systems, stubs control the transformation of data formats.^{11,12}

Automatic stub generation. The client and server stubs are normal modules, linked to the client and server respectively and forming part of their task. They could be written by hand. However, it is much more convenient and safer to generate them automatically with a stub generator.¹³ The RPCGEN stub generator analyzes the file that defines the server module. This server definition file is, of course, common to the client and server modules. RPCGEN generates the implementation modules of both the client and server stubs. The principle of stub generation is shown in Figure 7.

Each client receives a copy of the client stub.

The stub generator also examines the symbol file generated by the compiler to detect version conflicts. The symbol file contains a version key, which the stub transmits with each message to enforce type checking over the network.

Service interface language

All RPC systems rely on a server definition file. It is called the network interface definition file in Apollo's NCS and the I/F specification in Delta-4.

The server definition file could be written in a standard computer language, like a definition file in Modula-2. Although simple applications could run with some default settings, this solution is not general enough. The stub generator needs additional informa-

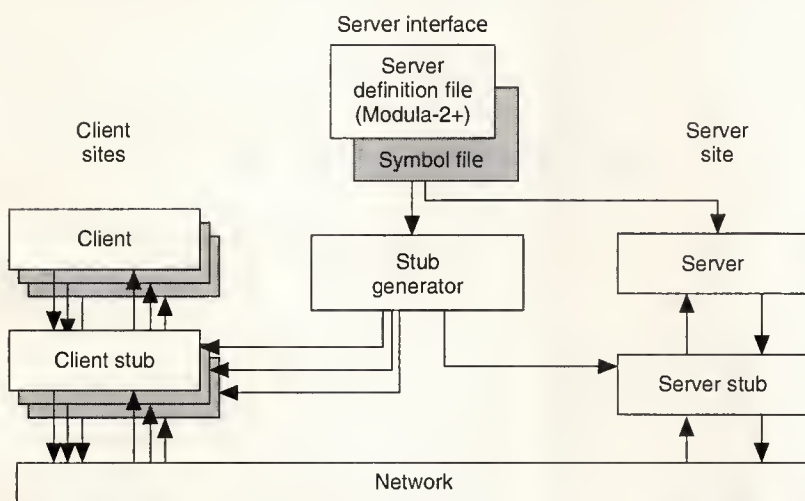


Figure 7. Stub generation.

tion which is not relevant when a procedure is called directly from within the same task. For instance:

- *Procedure type.* Remote procedure calls (blocking) and remote procedure invocations (nonblocking) are distinguished by an "RPC" or "RPI" prefix.
- *Procedure parameters.* Since clients and servers do not share a common address space, the stubs copy all parameters, even those passed by reference (pointer). To avoid unnecessary copying, we distinguish parameters that the procedure may only read (input parameters, or "arguments"), parameters the procedure may modify (output parameters, or "results"), and parameters the procedure both reads and writes (input/output parameters). Ada is one of the few languages that tags the parameters as IN, OUT, or INOUT. We use the same convention in Alphorn.
- *Dynamic-size parameters.* The size of this type of parameters is not part of the interface, since it is known only at runtime. A dynamic array declaration allows us to copy these parameters correctly.
- *Parallelism.* The degree of parallelism (that is, the number of server threads) and the degree of fault tolerance (the number of replicates) can be indicated in the server definition file. This data can be modified at configuration time.
- *Presentation.* A description of the target machine is included in the interface definition so that the stubs can convert types automatically. Since the data type is part of the server interface, the client must adapt. For instance, a client stub converts 32-bit integers returned by a little-endian (Intel) server to its own big-endian (Motorola) format. Since the parties agree on the data representation at compile time, there is no need for an on-line negotiation of data types. This avoids a presentation layer in the communication process. A unified data type presentation can also be specified, similar to Sun's XDR.⁵
- *Initialization.* Some clients come into existence when the system is started. These "top actions" are specified as services called from nowhere.

The server interface is expressed in a service interface language. ECMA, ROSE (Remote Operations Service Elements, ISO 9072), and Delta-4 use the standardized Abstract Syntax Notation 1 (ASN1, ISO 8824) to achieve independence of any computer language. However, ANSA and IEC SC21 consider ASN1 suitable for expressing message structures but too weak as an interface description language; they have developed their own interface description language, derived from Xerox's Courier protocol.¹⁴

For its service interface language, Alphorn uses the syntax of a normal Modula-2 definition file, in which the stub control instructions are enclosed in comments. This allows features not found in ASN1. The main advantage is that the

Modula-2 compiler can process the file directly. This is all that is needed to run within a single task. To make a module remotely accessible, the stub generator analyzes the stub control information enclosed in the comments and the symbol file (to retrieve the version key).

The use of Modula-2 has an interesting by-product: Modula-2, like most languages, was created as a sequential language and not to support parallelism. As such, it defines a static structure. Clients and servers, however, are dynamic entities. The Modula-2 syntax is used here to express a dynamic structure, which becomes apparent only at runtime. This interpretation is done by the stub generator.

Configuration

Once the servers, the top client modules, and the server stubs have been compiled, they are configured; that is, the entities belonging to the same site are linked together to form one runnable task. There may be several tasks in the same node, each one comprising servers and top modules. Server modules are not active by themselves; they need a client to call them. Top actions become active when the task is started. When they call a service, they become top clients. A task is formed by a main Modula program that directly imports the top modules and the server stub modules (Figure 8). The top modules import client stubs as needed, while the server stub modules import their corresponding server modules.

The main program provides the Runtime Support. (RTS is similar to Deltase in the Delta-4 project and to the ANSA nucleus.) It imports both the top clients and the server stubs. The main program creates and manages a number of threads. Conceptually, all these threads are parallel—that is, they would execute in parallel if there were a sufficient number of processors. They form a pool that runs the servers and top actions. The runtime support provides the communication interface.

The writing of the main program is automated by the OBJGEN utility. The OBJGEN configurator generates a main program, which is a normal Modula-2 program, to hold the top client, client stubs, server stubs, and server modules. OBJGEN interprets the definition files as follows:

- For each procedure (without parameter) exported by a top module, a thread is created and started. These threads form the top actions. Conceptually, all top actions are started simultaneously. A top action becomes a top client when making a remote call.
- For each server stub, several server threads, or servers for short, are created. When an action calls a service, one of these servers will execute the corresponding procedure as a server action. The number of server threads can be specified in the definition file during stub generation. The default value is 3 in the present implementation.

Runtime and debugging tools

A running system consists of a number of nodes, each running several application and system tasks. Network communication is only partially realized in the stubs. Other entities must exist to support them, in particular the networker. The network configuration is shown in Figure 9 on the next page.

Each task may contain servers and top actions. Top actions usually call services of other servers, but they need not. For instance, in Figure 9, Task 2, the top action of node Alpha, needs no access to a stub since it performs no remote calls.

Networker. A stub performs only part of the communication work. Basically, it provides the glue between the application interfaces and the standard low-level RPC mechanism. A driver called the networker executes the actual communication. The networker is permanently active; that is, it is a separate task or a device driver (except on single-task systems like MS-DOS, where it is linked to the application). This driver is common to all applications in a node, and it hides the details of communication from the stubs.

The networker executes the RPC protocols. It analyzes the available communication links—for instance, the shared-memory communication module, the Ethernet driver, DECnet to access another computer, or X.25. When the networker recognizes that client and server are in the same node, it connects the stubs directly through shared memory. Similarly, when a DECnet link exists between client and server, the networker uses its facilities directly.

Name server. When a client calls a server in the network, its networker must know the location of this server. That could be specified during configuration by early binding. Alphorn includes and removes separately compiled modules as the system runs. It uses late binding to indicate the location of the servers at runtime. Each networker performs the name server function—there is no central trader. The name server publishes all services in the network. (It does not publish their interfaces, however, unlike the Delta-4 trader. That would have required each node to have mass storage.) A multicast protocol continually actualizes all name servers.

The same service may be installed several times in the system, to increase performance or fault tolerance. In the first case, the user may wish to choose and distinguish which server is processing the call. This is especially important for exception handling. Thus, the name server provides a mechanism to

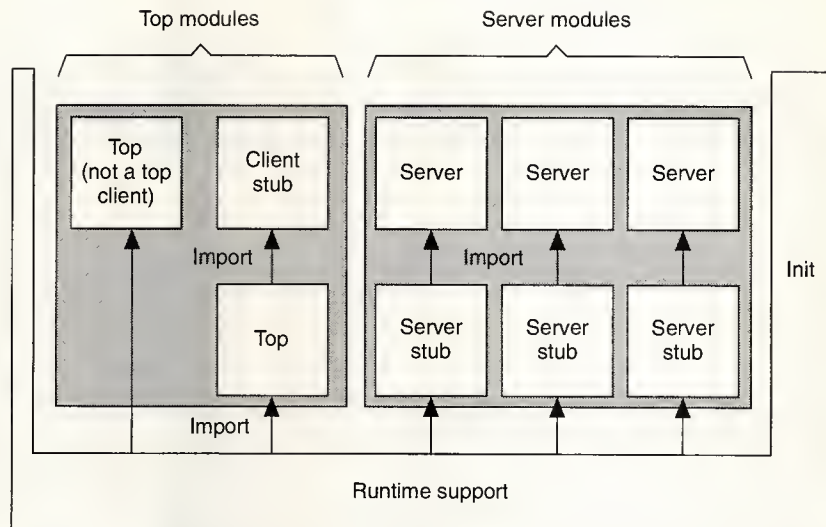


Figure 8. Configuring a task with top and server modules.

identify the client and the server that participate in a call. When a service is replicated for fault tolerance, both the primary and the backup carry the same identification. The networker monitors the start and termination of server and clients, notifying all interested applications.

NetMonitor. During development and test, it is useful to have an insight on the way communication takes place and which entities are active at a time. The NetMonitor utility runs as a task anywhere in the network. It monitors existing sites, traces established connections, lists clients, servers, handlers, and signalers, and displays statistical information. Finally, NetMonitor can stop sites in an orderly way.

Fault tolerance support

There is no general-purpose fault-tolerant computer. To handle redundancy effectively, a fault-tolerant system must take account of the plant or other process control application. Fault tolerance therefore requires close collaboration between the application programmer and the system designer. Alphorn provides the following basic tools to help them construct fault-tolerant systems:

Exception handling. As is not the case with a local call, one must be prepared for the failure of a remote call. To maintain the same interface as for local calls, a call's status is delivered by a "hidden" module called RPC Status. Since the client is blocked during the call, a failure of the server must be signaled by an exception handler. This is a critical component, since some machines and languages support exception handling well, such as VAX/VMS, while others seem to ignore the problem. Much of the porting work for Alphorn consisted of implementing exception handlers for machines that lacked them.

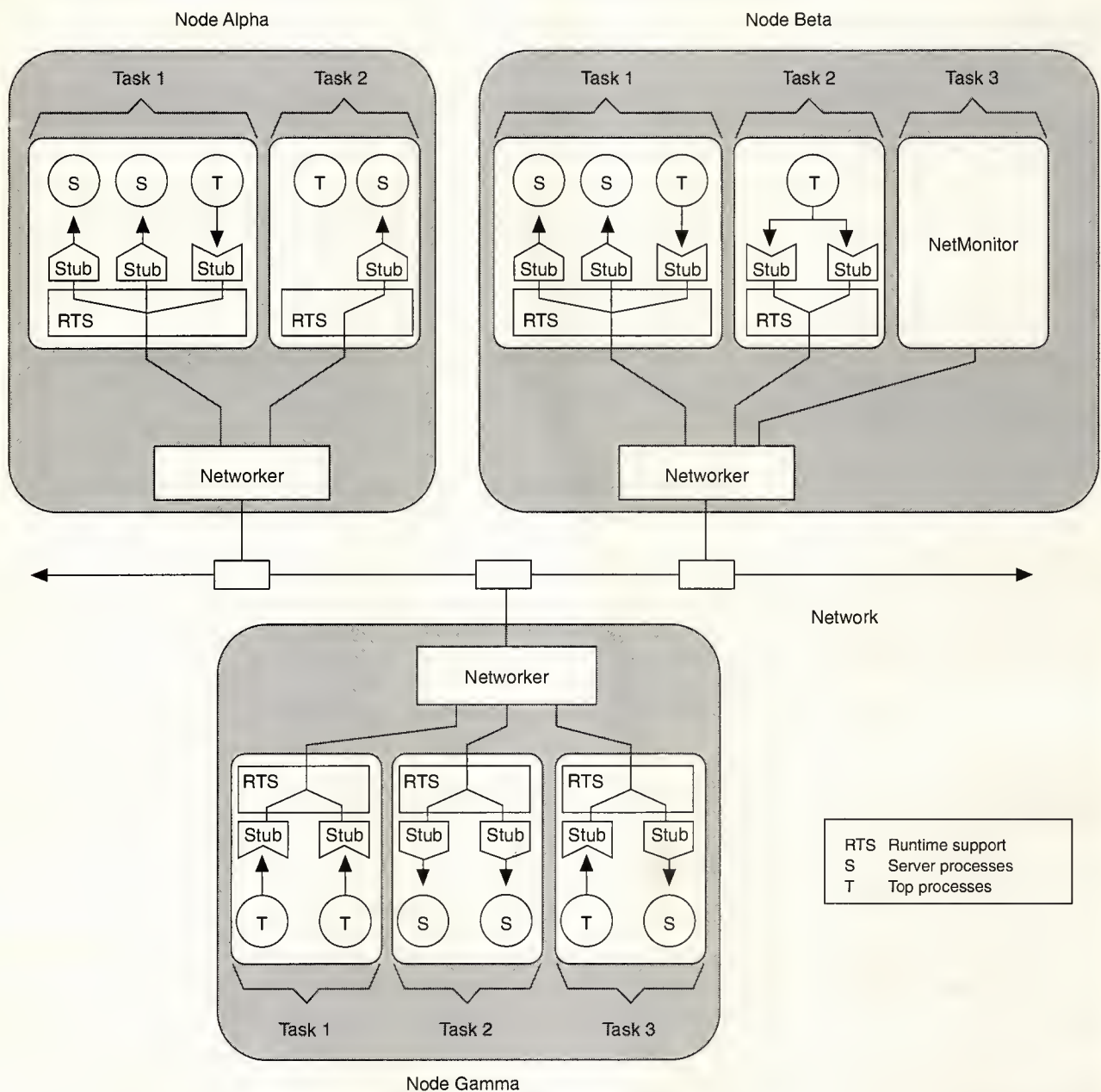


Figure 9. Network at runtime.

Error reporting. Each node constantly monitors the network. Healthy nodes regularly communicate or send "I'm alive" messages; the absence of messages triggers reconfiguration. Therefore, the worst-case error detection latency is equal to the network's cycle time. This mechanism is part of the basic runtime support.

Redundant distributed system. In process control at the supervisory level, the main requirement is high availability. Basic availability is best provided by a duplex structure. Functional redundancy is provided by additional nodes in the network and duplication of communication links. When a node fails, its backup node takes over its function. The backup



October 1991 issue (card void after April 1992)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest

(Add comments on the back)

Readers,
Indicate your interest in
articles and departments
by circling the appropriate
number (shown on
the last page of articles
and departments):

150 151 152 180 181 182
153 154 155 183 184 185
156 157 158 186 187 188

159 160 161 189 190 191
162 163 164 192 193 194
165 166 167 195 196 197

168 169 170 198 199 200
171 172 173 201 202 203
174 175 176 204 205 206

177 178 179 207 208 209

Product Information 1

(Circle the numbers to receive
product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	



October 1991 issue (card void after April 1992)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest

(Add comments on the back)

Readers,
Indicate your interest in
articles and departments
by circling the appropriate
number (shown on
the last page of articles
and departments):

150 151 152 180 181 182
153 154 155 183 184 185
156 157 158 186 187 188

159 160 161 189 190 191
162 163 164 192 193 194
165 166 167 195 196 197

168 169 170 198 199 200
171 172 173 201 202 203
174 175 176 204 205 206

177 178 179 207 208 209

Product Information 2

(Circle the numbers to receive
product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	

SUBSCRIBE TO IEEE MICRO

All the facts about today's chips and systems

☐ **YES**, sign me up!

If you are a member of the Computer Society or any other IEEE society,
pay the member rate of only \$21 for a year's subscription (six issues).

Society: _____

IEEE membership no: _____

Society members: Subscriptions are annualized. For orders submitted March through
August, pay half the full-year rate (\$10.50) for three bimonthly issues.

Full Signature _____ Date _____

Name _____

Street _____

City _____

State/Country _____ ZIP/Postal Code _____

☐ **YES**, sign me up!

If you are a member of ACM, ACS, BCS, IEE (UK), IEEE but not a
member of an IEEE society), IECEJ, IPSJ, NSPE, SCS, or other
professional society, pay the sister-society rate of only \$38 for a year's
subscription (six issues).

Organization: _____

Membership no: _____

☐ Payment enclosed *DC residents: add sales tax*
☐ Charge to ☐ Visa ☐ MasterCard ☐ American Express
Charge-card number _____

Expiration date _____

Month _____ Year _____

Prices valid through 12/31/91
10/91 MICRO

Charge orders also taken by phone:
(714) 821-8380 8 a.m. to 5 p.m. Pacific time
Circulation Dept.
10662 Los Vaqueros Circ., PO Box 3014
Los Alamitos, CA 90720-1264

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

Reviewers Needed. If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.

For reader-service inquiries, see other side.

PO Box is for reader-service cards only.

PLACE
POSTAGE
HERE

IEEE Micro

PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA



Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

Reviewers Needed. If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.

For reader-service inquiries, see other side.

PO Box is for reader-service cards only.

PLACE
POSTAGE
HERE

IEEE Micro

PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

IEEE COMPUTER SOCIETY

PO BOX 3014
LOS ALAMITOS CA 90720-9804
USA

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



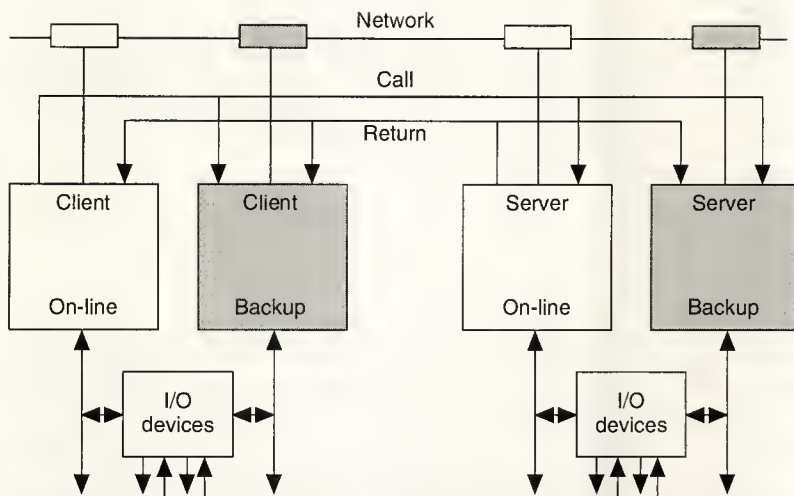


Figure 10. Resilient remote procedure calls.

node is functionally redundant to the on-line node. In particular, the backup has the same access to I/O devices as the on-line node, either through dual-ported devices or redundant devices.

Not all nodes need to be duplicated. The Alphorn fault tolerance concept supports both replicated and nonreplicated nodes in the same system. Higher levels of replication such as triplication (two backups) may also be integrated. Most fault-tolerant applications in Alphorn, however, are expected to be of the duplex type.

Error detection. The failure of a node is detected by means such as parity checks, self-checking circuits, watchdog timers, and the like. We assume that each node is fail-stop—it will stop sending data in case of failure (“fail-silent” in Delta-4). For process control at the supervisory level, integrity is not critical and the self-checking provided by commercial mainframes is generally sufficient. Full integrity—no false data in case of failure—is mandatory in critical applications such as railway signaling. In such cases, each node must be duplicated to provide sufficient coverage.

It makes no sense to increase fault detection coverage of hardware past a certain point. Software causes many errors, which can hardly be caught by hardware means. That is why Alphorn provides no protection against malicious behavior or babbling nodes. We do not try to provide tolerance of software errors, even in nonredundant cases, so that the failure of one unit will not cause others to crash. For this purpose, Alphorn relies on its exception-handling mechanism.

Redundancy actualization. It is not enough to add redundant nodes to a network. Unless the state of backup nodes is close enough to that of their on-line units, switchover will be rough or even impossible. There are two basic techniques

to keep backup computers actualized: standby, or periodical update, and workby, or parallel operation (called passive and active replicates in Delta-4; coordinated and parallel replica in ANSA).¹⁵

- In the standby (or asynchronous) mode, a backup node is regularly actualized by transfers from the on-line node. These state transfers take place at checkpoints. Otherwise, the backup is free to perform other tasks.
- In the workby (or synchronous) mode, the on-line node and the backup node perform the same tasks at the same time. Therefore, their internal states remain identical. In principle, their outputs could be compared to detect errors. The backup cannot be used for other tasks.

Workby requires synchronization and matching to remove all sources of nondeterminism, which could let the units diverge. Synchronization and matching over the network costs time.¹⁶

Alphorn supports both operation modes, but unlike ANSA and Delta-4, Alphorn supports no replica groups. In either mode, the backup must constantly monitor the network activity to reproduce or coexecute the actions of the primary. In both cases, a teaching mechanism updates a fresh node to backup status.

Resilient remote procedure calls. Standard RPCs can be easily extended to support both the standby and the workby modes of operation.¹⁷ The basic idea is that actualization of the backup, whether through checkpointing or synchronization, requires that communication over the network be monitored. To this effect, both the on-line nodes and their backups receive call and return messages (Figure 10).

For instance, when a call is made, it is received by the server and by the backup server. If the server fails, the backup will have received the service call and can execute it. Or, if the client fails, its backup can receive the return message in its place and proceed. Thus, each RPC becomes an implicit checkpoint. This method effectively eliminates the domino effect.¹⁸

Causal broadcast. Actualization and synchronization of replicates over the network require reliable multicast communication. (Although some local networks provide acknowledged broadcast transmission, this property cannot be expected of open networks.) Rather than a full-fledged atomic broadcast protocol, Alphorn relies on a causal broadcast protocol.¹⁹ A causal broadcast protocol ensures that message order is maintained under all circumstances for all interested parties. This protocol is built on top of existing protocols such as

**Swiss PTT uses Alphorn
for a distributed database
of 200 nodes scattered
over the country.**

DECnet or X.25. Causal broadcast has proved to be very efficient and handy, even in nonreplicated applications. (Unlike Delta-4, this protocol does not provide authentication or signatures; protection against intrusion is unnecessary in a dedicated system.)

Application dependence. Ideally, the application programmer writes programs without caring about replication. It is indeed desirable to offer, for instance, a control system in a redundant and in a nonredundant configuration. Therefore, the code running in the on-line unit and in the backup unit must be identical. Redundancy is introduced at configuration time and hidden in the runtime system or in the device drivers. This ideal cannot be completely maintained.

Fault tolerance constructs. Alphorn provides useful constructs to support fault tolerance. While each RPC provides a synchronization/rollback point, it may be necessary to include checkpoints at closer intervals, especially for top actions, which are never called as server. To this effect, Alphorn provides an iteration construct, which inserts a checkpoint each time a loop is executed.

In standby mode, data modified in the primary node are transmitted to the backup unit. Indiscriminate checkpointing, as used in transaction computers, is not acceptable in real-time systems. Alphorn provides a construct for tagging data structures that are subject to modification. Only these data will be transmitted to the standby when the call returns.

This checkpointing allows us to build atomic actions, which are either completely executed or not at all. When a failure occurs, computation starts from the last checkpoint (rollback) and proceeds to repeat operations already done by the failed unit (rollahead), until reaching the failure point. The rollahead procedure avoids repeating output operations already performed by the failed unit. But in some situations, it may be necessary to redo such operations—for example, refreshing the operator's screen. Here, the application dependency must be hidden in the I/O drivers. Alphorn does not, however, provide atomic transactions; this is considered an application issue.

Teaching backups. The teaching of new backups—for instance, after repair—is performed in the background and consists of a dialogue-like communication between an on-line task and the future standby task, to transmit the static

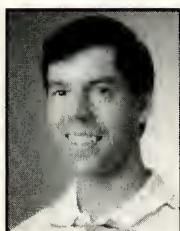
state of an object. Depending on the amount of spare computing time, two options are available: concurrent teaching goes on during normal operation but increases load. Off-line teaching requires a small interruption but costs no runtime overhead.

THE ORIGINAL GOAL OF THE ALPHORN PROJECT was to develop a fault-tolerant computer network. Later, the software environment and the network-independent communication based on RPC became the mainstream of the project. Although based on Modula-2, Alphorn is not limited to that language. Stubs may be easily generated for other languages, such as C. In fact, the target language, machine, and compiler are options of the stub generator. A Pascal version has been developed for a machine that has no Modula-2 compiler. In collaboration with the Swiss Federal Institute of Technology, in Lausanne, we are working to provide manufacturing message services (ISO 9506) as Alphorn services. Alphorn is now used for a distributed database implemented on behalf of the Swiss PTT (post office), consisting of 200 nodes scattered over Switzerland. That is in keeping with the project's name: alphorns are traditional Swiss musical instruments derived from the long horns shepherds once used to call their herds across the valleys. ■

References

1. H. Kopetz and W. Merker, "The Architecture of MARS," *Proc. 15th Symp. Fault-Tolerant Computing*, June 1985, pp. 274-279.
2. N. Wirth, *Programming in Modula-2*, Springer-Verlag, 1982.
3. K. Birman and T. Joseph, "Reliable Communication in the Presence of Failures," *ACM Trans. Computer Systems*, Vol. 5, No. 1, Feb. 1987, pp. 47-76.
4. "Network Computing System (NCS) Reference Manual," Order No. 010200, Hewlett-Packard/Apollo Computer, Inc., Chelmsford, MA 01824, June 1987.
5. "Network Programming Guide," Part No. 800-3850-10, Sun Microsystems, Inc., Mountain View, Calif., 1990.
6. "ANSA Advanced Networked Systems Architecture, An Engineer's Introduction to the Architecture," Architecture Projects Management Ltd., Poseidon House, Castle Park, Cambridge CB3 0RD, UK, 1989.
7. D. Powell et al., "The Delta-4 Distributed Fault-Tolerant Architecture," LAAS Report No 91055, LAAS, 7 Avenue du Colonel Roche, F-31077 Toulouse, France.

8. ECMA-127, "Basic Remote Procedure Call Using OSI Remote Operations," final draft, European Computer Manufacturers Association, 114 Rue du Rhone, CH-1204 Geneva, Switzerland, Jan. 1990.
9. B. Liskov, "The Argus Language and System," in *Advanced Course on Distributed Systems*, Springer-Verlag, Munich, 1984.
10. H.R. Aschmann, "Resilient Remote Procedure Call," Research Report CRB 88-09 C, Asea Brown Boveri Research Center, CH-5405 Dattwil, Switzerland.
11. P.B. Gibbons, "A Stub Generator for Multilanguage RPC in Heterogeneous Environments," *IEEE Trans. Software Eng.*, Vol. SE-13, No. 1, Jan. 1987, pp. 77-87.
12. B. Bershad et al., "A Remote Procedure Call Facility for Interconnecting Heterogeneous Computer Systems," *IEEE Trans. Software Eng.*, Vol. SE-13, No. 8, Aug. 1987, pp. 880-894.
13. A. Birrell and B. Nelson, "Implementing Remote Procedure Calls," *ACM Trans. Computer Systems*, Vol. 2, No. 1, Feb. 1984, pp. 39-59.
14. "Courier: The Remote Procedure Call Protocol," Xerox System Integration Standard, Xerox Corp., Stamford, Conn., Dec. 1981.
15. H.D. Kirmann, "Fault Tolerance in Process Control," *IEEE Micro*, Vol. 7, No. 5, Oct. 1987, pp. 27-50.
16. E.C. Cooper, "Circus: a Replicated Procedure Call Facility," *Proc. Fourth Symp. Reliability in Distributed Software and Database Systems*, Oct. 1984, pp. 11-24.
17. H.R. Aschmann, "Broadcast Remote Procedure Calls for Resilient Computation," *Proc. IFAC Safecomp*, 1985, pp. 109-115.
18. B. Randell, P.A. Lee, and P.C. Treleaven, "Reliability Issues in Computing System Design," *ACM Computing Surveys*, Vol. 10, No. 2, June 1978, pp. 123-165.



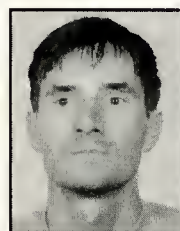
Aschmann



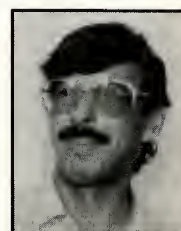
Hoepli



Giger



Janak



Kirmann

Hans-Ruedi Aschmann is a project leader in data communications with Glance Ltd. in Steinmaur, Switzerland. He received the BS degree in electronics and the PhD degree from the Swiss Federal Institute of Technology in Zurich after postgraduate studies at Rensselaer Polytechnic Institute in Troy, New York.

Elisabeth Hoepli works as an informatic engineer for the Asea Brown Boveri Research Center in Baden. Her technical interests include real-time software and data integrity in communication systems. Hoepli received her BS degree in physics from the Swiss Federal Institute of Technology.

Niklaus Giger works as a consultant for the Asea Brown Boveri Research Center in Baden. He received his BS degree in electronics.

Peter Janak, a software engineer, leads the Non-Stop Laboratory at the Cantonal Hospital of Saint Gall, Switzerland, for ABB Corporate Research. After his studies at the Interstate

College of Buchs, Switzerland, he pursued postgraduate studies at the Swiss Software School in Bern.

Hubert Kirmann leads the Computer Architecture Group at the Asea Brown Boveri Research Center. Previously, he taught courses in automatic control at the District University of Bogota, Colombia, and developed electrical sensors. Kirmann received his BS degree in electronics and his PhD degree from the Swiss Federal Institute of Technology. He is a member of the editorial board of *IEEE Micro*.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 165

Medium 166

High 167

RISC processor

continued from p. 23

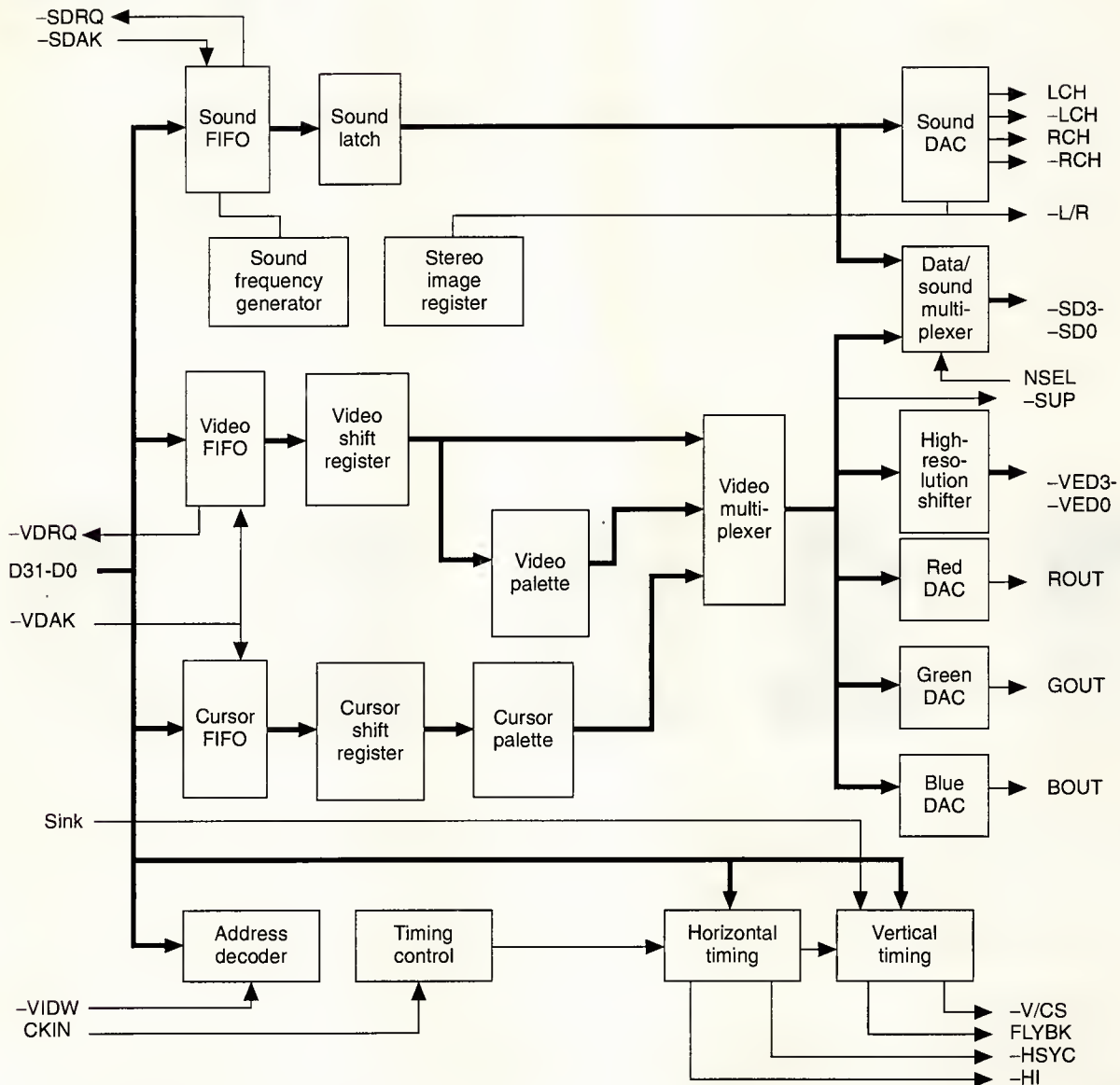


Figure 4. VL86C310 VIDC block diagram.

cluding the digital-to-analog converters for red, green, blue, and sound. (See Figure 4.)

The VIDC supports pixel rates from 8 to 24 MHz, data serializing from 1 to 8 bits per pixel, fully programmable screen parameters, and flexible cursor sprites.

I/O controller

The VLSI VL86C410 I/O controller (IOC) shown in Figure 5 provides a unified environment for I/O related activities such as interrupt and peripheral controllers.

The controller contains elements such as timers, baud rate

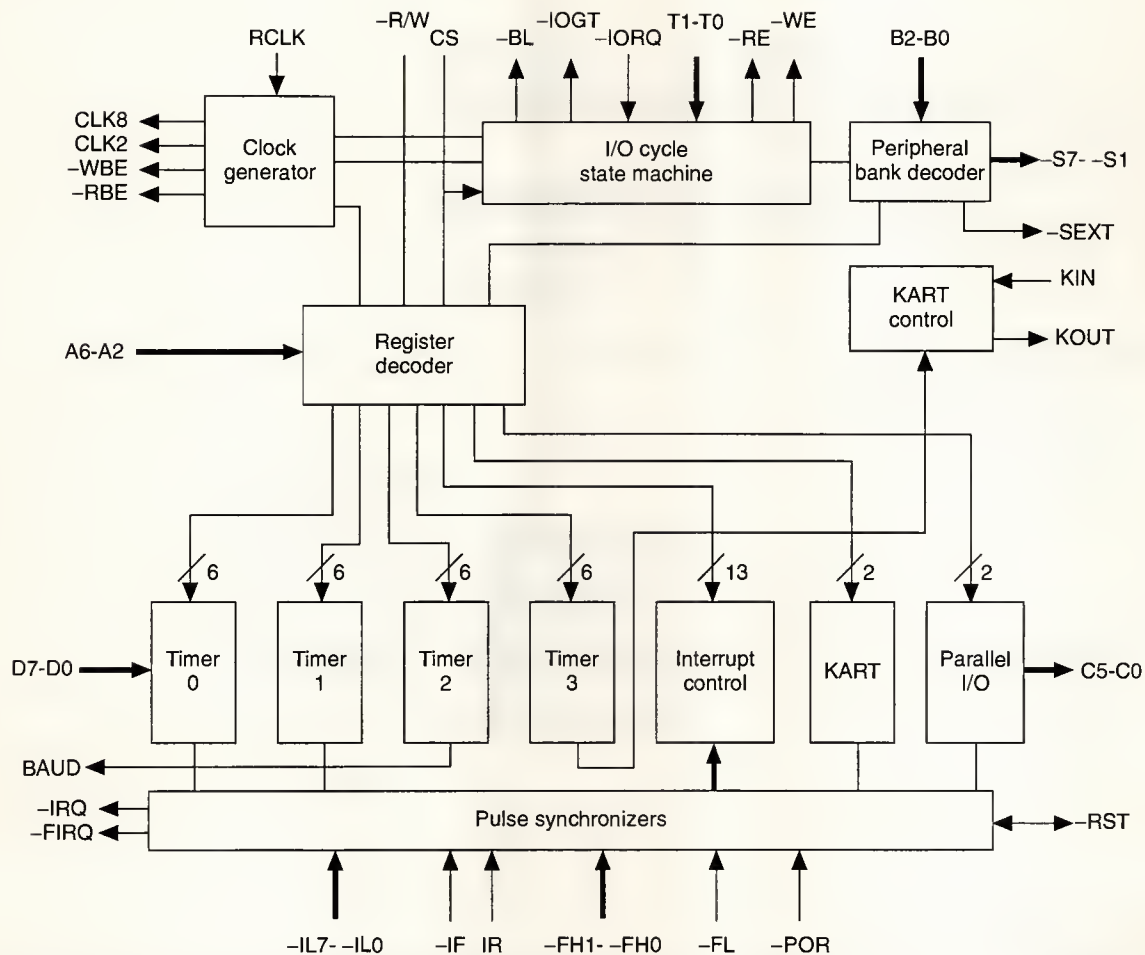


Figure 5. VL86C410 IOC block diagram.

generators, keyboard controller, and parallel I/O port control. Again, the IOC was designed as part of a system to maximize the performance/cost ratio. Only part of the data and address buses connect to the IOC. This small pin count helps in ASIC applications by reducing busing on the ASIC chip.

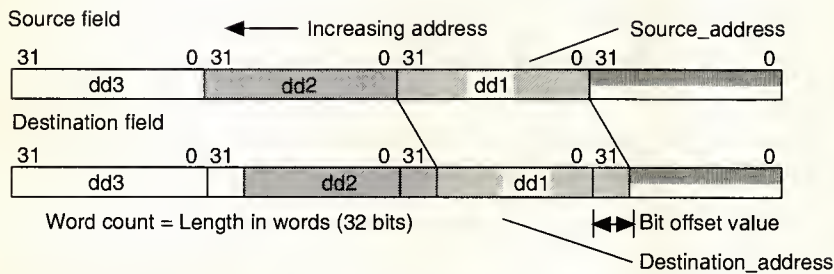
Performance

Let's look at the VL86C010 performance rate while keeping in mind one caveat: Benchmarks should always be looked at with caution, even if you have created them yourself. Unless the benchmark is the actual debugged application code, it can be inapplicable or can overlook some operational detail that arises when using the code in a particular application.

A published benchmark² for the Intel 386 family for bit-block graphics shows a final drawing rate of 17.067 million pixels per second at a processor clock rate of 16 MHz (see

Figure 6a on the next page). The VL86C010 performs the same function (using different instructions, of course) with a processor clock of 12 MHz and achieves a drawing rate of 30.77 million pixels per second (see Figure 6b).

Another example (shown in Figure 7) involves an MC68020 coded to solve $z = 2x + y$, where x and y are variables that could not be destroyed. The example was coded two different ways with the variables in registers (already loaded) as seen in Figure 7a and the variables in a stack (stack pointer initialized) in Figure 7b. The first example took the Motorola processor eight clock cycles and the VL86C010 one clock cycle. The second example took the 020 processor 18 clock cycles (Figure 7c) and the VL86C010 nine clock cycles (Figure 7d). A final example (see Figure 8) demonstrates the effectiveness of our RISC's architecture and conditional execution compared to that of the MC68020.



Intel 80376 Example Code - 64-bit barrel shifter and 16-bit data bus

```
MOV ESI, Source_address
MOV EDI, Destination_address
MOV EBX, Word_count
MOV CL, Relative_offset
MOV EDX, [ESI]
ADD ESI, 4
```



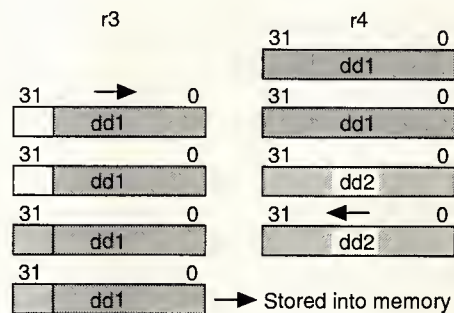
	Clock cycles	Length (bytes)
BitLoop: LODSD	7	1
SHRD EDX, EAX, CL	3	3
XCHG EDX, EAX	3	1
STOSD	6	1
DEC EBX	2	1
JNZ BitLoop	9	2
Totals	30	9

(a)

;VL86C010 register usage

```
;r1 - Source_address      r3 - dd1 data
;r2 - Destination_address r5 - Relative_offset
;r4 - dd2 data             r7 - Word_count
;r6 - 32 - Relative_offset
```

```
LDR r1, Source_address
LDR r2, Destination_address
LDR r5, Relative_offset
LDR r4, [r1]+
RSB r6, r5, 32
```



		Clock cycles	Length (bytes)
BlitLoop:	MOV r3, r4 LSR r5	2	4
	LDR r4, [r1]+	3	4
	ORR r3, r3, r4 LSL r6	2	4
	STR r3, [r2]+	2	4
	SUB r7, 1	1	4
	BNE BlitLoop	3	4
	Totals	13	24

(b)

Figure 6. Bitblt example in graphics application: Intel 376 family at 16-MHz drawing rate of 17.067 Mpixels/second (a) and the VLSI Technology VL86C010 processor at the 12-MHz drawing rate of 30.77 Mpixels/second (b).

Calculate $z = 2x + y$ where x and y are register variables that cannot be destroyed.

```
move.l  rx, rz    ; (2) rz=x
lsl.l   #1, rz    ; (4) rz=2x
add.l   ry, rz    ; (2) rz=2x+y
```

Requires three 16-bit instructions and eight clock cycles (cache case).

(a)

```
add     rz, ry, rx, lsl #1    ; (1) rz=2x+y
```

Requires one 32-bit instruction and one clock cycle.

(b)

Using x and y as stack variables (stack pointer initialized),

```
move    offx(a6), rz        ; (7) z=x
lsl.l   #1, rz              ; (4) z=2x
add.l   rz, offy(a6)        ; (7) z=2x+y
```

Requires two 32-bit, one 16-bit instruction, and 18 clock cycles.

(c)

```
ldr     rtmp1, offx(r13)    ; (4) tmp1=x
ldr     rtmp2, offy(r13)    ; (4) tmp2=y
add     rz, rtmp2, rtmp1 lsl #1 ; (1) z=2x+y
```

Requires three 32-bit instructions and nine clock cycles.

(d)

Figure 7. Code examples: with the variables in registers for the MC68020 (a), for the VL86C010 (b), with the variables in a stack for the MC68020 (c), and the same for the VL86C010 (d).

A product growth path is essential to ensure continued speed and applicability. The performance figures generally given for most RISC processors are based on the use of static RAMs with state-of-the-art access times. For these processors to double their performance, the RAMs must perform twice as fast, and this follows right on up the performance curve. I have quoted VL86C010 performances based on the use of 80-ns DRAMs, which are slower than the SRAMs.

The newest member of the family, the VL86C020, contains an on-chip cache and produces 2.5 times the performance using the same DRAMs. Additionally, if an application warrants it, the processor can be run faster using static RAMs. These CPUs have a robust growth path within the existing memory technology.

VIRTUALLY ALL RISC CPUs ON THE MARKET TODAY have die sizes that are at the upper limit of that which is economically producible. This die size rules out their use as

Logical Or function:

If $R_n = p$ OR $R_m = p$, GOTO LABEL

```
cmp     Rn, #p    ; (2) If Rn=p OR Rm-q then
beq     LABEL    ; (6) GOTO LABEL
cmp     Rm, #q    ; (2)
beq     LABEL    ; (6)
```

Requires two 16-bit instructions and two 48-bit instructions when using long addressing (32-bit displacement) and 16 clock cycles when cache-resident and the condition is met by the second case.

(a)

```
cmp     Rn, #p    ; (1)
cmpne   Rm, #q    ; (1) if Rn not equal p, try other
; test
beq     LABEL    ; (4)
```

Requires three 32-bit instructions and six clock cycles for the worst case in either case.

(b)

Figure 8. Code examples: Logical Or function for the MC68020 (a) and for the VL86C010 using conditional execution (b).

cores in the design of other ICs. The VL86C010 RISC CPU, on the other hand, measures approximately 30 percent of the size of other RISC CPUs. In addition, the busing scheme of the entire family allows efficient designs in ASICs. In fact, our designers implemented the new VL86C020 RISC CPU with on-chip cache memory by placing the 010 core on a cell-based chip and then adding the RAM and control logic to form the cache.

Certainly, if one is designing an embedded controller today that needs RISC performance, small size, and low power and cost with increased reliability and security, there exists a way to do it efficiently with ASICs. ■

References

1. Acorn RISC Machine Family Data Manual, Prentice Hall, Inc., Englewood Cliffs, N. J., 1990.
2. C. Purkiser and J. Kardach, "The Intel 376 Family for Embedded Processor Applications," *IEEE Micro*, Vol. 8, No. 3, June 1988, pp. 10-26.



Charles E. Roberts is VLSI Technology, Inc.'s senior field applications engineer for the southeastern United States. He has been involved in microprocessor-based product design for the past 18 years and holds a patent in the field. He studied computer science engineering at the University of South Florida and is a voting member of the Association of Computing Machinery.

Direct any questions about this article to the author at VLSI Technology, Inc., 2200 Park Central North, Suite 600, Pompano Beach, FL 33064.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 162

Medium 163

High 164

1992 PUBLICATIONS CATALOG

IEEE COMPUTER SOCIETY
PRESS

Contains:
16 New Computer Science
Books,
9 New Videotapes,
8 New Standards,
and
New 1991 Proceedings !

Call 1-800-CS-BOOKS
for your copy today

1951-1991
40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

i860 performance

continued from p. 27

lines can be cascaded to become six stages (Figure 5). The i860 defines 32 different dual-operation opcodes, allowing add-before-multiply, multiply-before-add, and other combinations. Three 64-bit registers, called KR (constant real), KI (constant imaginary), and T (temporary), hold results between the adder and multiplier.

The other parallel feature is simultaneous execution of both an integer and floating-point instruction in dual-instruction mode (DIM). In this mode, explicitly controlled by the programmer, the instruction cache fetches two instructions at once, one for the floating-point unit and the other for the integer unit (Figure 6). Designers created the dual-instruction mode to keep the floating-point unit (and graphics) running at full bandwidth, in spite of the need for instructions to

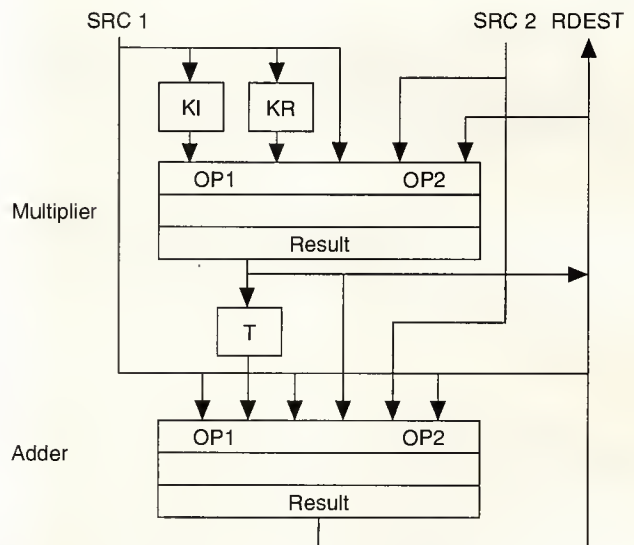


Figure 5. Floating-point adder and multiplier paths.

D.PFADD	F	D	E1	W _E	E3		
ADDU	F	D	E	W			
D.PFSUB		F	D	E1	W _E	E3	
SUBU		F	D	E	W		
D.PFMUL			F	D	E1	W _E	E3
BR			F	D	I	W	
Clock cycle 0 1 2 3 4 5 6							

Figure 6. Dual-instruction-mode pipeline with two instructions each clock cycle.

move data between memory and registers. With the pipelined floating-point instructions producing one or two results every clock cycle, inserting the necessary load/store/branch instructions could halve the performance. Because the integer unit performs all memory accesses for the floating-point registers, it can accomplish data transfers without slowing the floating-point units. Dual-instruction mode is not needed with scalar-mode floating point, because two integer instructions can execute in the two clock cycles that occur between the initiation of consecutive floating-point scalar operations.

Dual-instruction mode parallelism is similar to superscalar implementations such as the IBM RS/6000 and Intel i960 processors. Superscalar hardware can determine at runtime opportunities for simultaneous execution of two or three instructions, without requiring explicit action by the programmer. However, the explicitly programmed dual-instruction mode is easier to use than VLIW (very long instruction word) machines that cause compilers difficulty in finding enough useful instructions to pack into a long-word unit of execution.

Programmers have put dual-instruction mode to effective use in the i860 software libraries for numerics and graphics. It doubles the performance of 3D rendering, fast Fourier transforms, matrix multiply operations, and other important kernels of code. Compilers will also exploit it, and future hardware can implement superscalar as well as DIM.

External pipeline. Like the internal instruction execution, the external bus of the i860 CPU exploits pipelining to increase bandwidth. Of course, the memory system can optionally ignore it for simplicity's sake, but most existing systems use this bus pipelining (Figure 7). They maintain the full bus bandwidth of 8 bytes transferred every two clock cycles, even though their access latency may rise to six cycles.

The NA# (next address) input pin and the ADS# (address strobe) output pin control the external pipeline. ADS# indicates a valid address on the bus, and NA# indicates willingness by external memory to accept another request, even though the data for the previous cycle has not yet been accessed. The CPU can issue up to three requests for reads and writes before the first data transfer completes. The READY# input to the CPU signals valid data on the data pins for reads, and that the memory has accepted the data on writes.

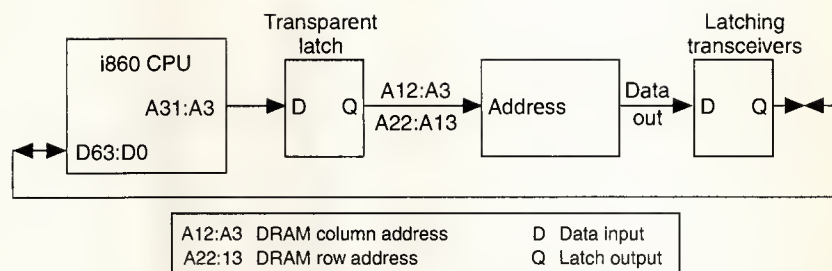


Figure 7. Typical external-memory pipeline hardware.

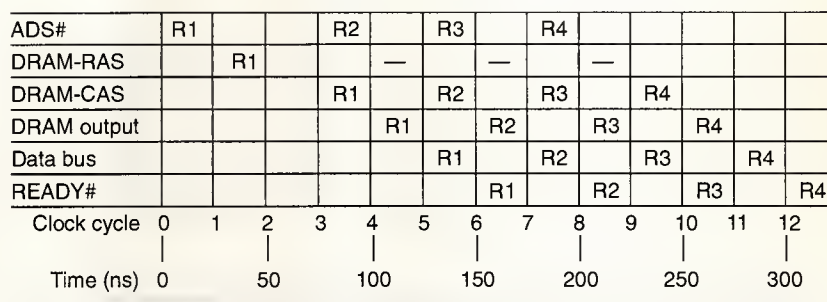


Figure 8. Pipelining for cache line fill. The four transfers, R1, R2, R3, and R4, each require two clock cycles at 40 MHz.

The on-chip caches have a line size of 32 bytes. With no external pipelining and a memory latency of six cycles, a cache line fill (four 8-byte fetches) takes 24 cycles to complete. However, by using external bus pipelining, a cache fill takes only 12 cycles with the same external memory latency, which represents a 50 percent faster cache fill.

External bus optimizations. The processor contains a comparator to drive an output pin called NENE# (next-near). This pin provides support for fast page mode and static column DRAMs. Such DRAMs can access data in the same page (typically 1 Kbit) in half the time required by access to a different page. For example, an 80-ns static column DRAM requires only 40 ns for a "near" read.

Actually, "next near" is a misnomer because the pin indicates previous near and is not capable of predicting the future. The NENE# pin indicates that the current memory access falls on the same DRAM page as the previous access. Consecutive accesses to the same page occur frequently. For example, cache line fills and write-backs (32 bytes) consist of three near accesses after the first access of the line. Figure 8 illustrates that RAS toggles only once for the four transfers of a cache line. To compensate for the inevitable increase in DRAM chip capacity, the page size used for NENE# derivation is software programmable in an on-chip control register.

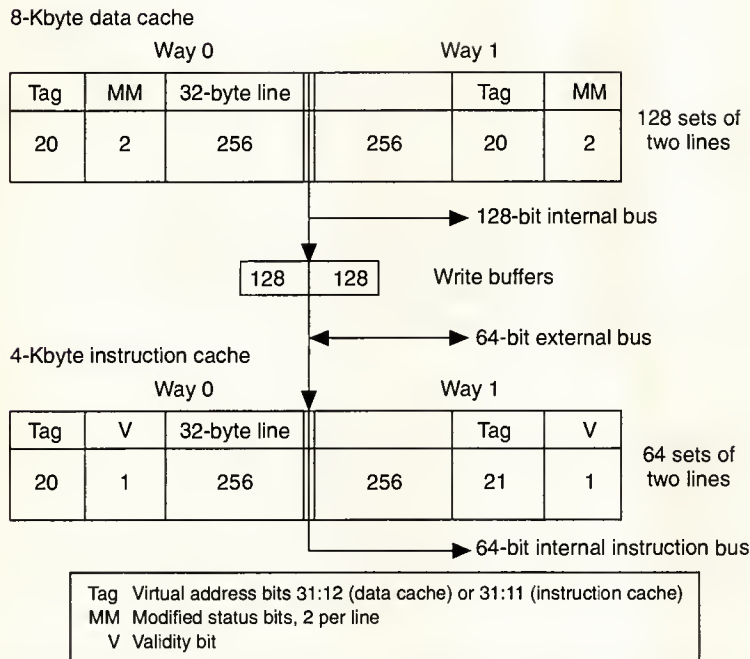


Figure 9. Internal caches with the widths of fields shown in bits.

Internal caches. The i860 processor uses split instruction and data caches (Figure 9) to ensure high bandwidth, as do the Mips R3000 and Motorola 88000 and 68040 processors. The 16-byte data path and 8-byte instruction bus at 40 MHz yield 960 Mbytes/s of access.

The feasibility of implementation on one chip in 1988 drove the design decisions of a 12-Kbyte size, two-way associativity, and replacement algorithm of the caches. Of course, the size and associativity are larger in the next-generation i860 XP CPU. To avoid the complexity required for least recently used (LRU) replacement, designers employed a pseudorandom algorithm to choose which of the ways to replace upon a miss. The high bandwidth and low latency (due to their locations on chip) offset the small size and replacement, as performance measurements show.

Virtual addressing. The virtually addressed caches require attention by operating systems programmers but are transparent to applications. A cache of this type uses a virtual address, rather than a real address, to select a line of data. Two advantages arise from virtual addressing in the hardware design:

- The cache access is very fast because the TLB (address translation look-aside buffer) lookup is not needed prior to the cache tag comparison. Translation of the physical address occurs in parallel with cache access and tag compare, not serially.

- The cache can grow in future implementations to greater than 4 Kbytes per way. Four Kbytes/way is the limit of most physical caches, because only the 12 least significant bits of address remain unchanged in the virtual-to-real translation with 4-Kbyte pages, which the i860, 386, 486, and most RISCs use.⁵ Physically addressed caches usually access the data and tag using those 12 bits, while the upper 20 bits of the address are undergoing translation in the translation look-aside buffer.

Virtually addressed caches contain addressing information that is obsolete whenever the operating system changes the translation tables. Thus the operating system must flush the caches during a context switch or when a memory page is swapped out to disk. This flush takes 50 μ s (or less—50 μ s is the worst case value when the entire cache is dirty, or modified) at 40 MHz, which decreases overall performance by 1 percent when

the CPU frequently switches contexts (250 per second).

Virtual tags also complicate aliasing of data that are writable in an associative cache. Aliasing in this sense means two different virtual address values refer to the same physical address, so that two copies of the same datum might appear in both ways of the two-way associative cache. For the case of a write to an aliased location, only one copy in the cache would be updated.

The second-generation i860 XP CPU executes hardware snooping.

Some operating systems use aliasing to share data or code between cooperating processes. Note, however, that read-only data and instructions can be aliased; if software requires writable data aliasing, it can make the cache behave as if it were a 4-Kbyte, direct-mapped cache by setting a bit in a control register and sacrificing some performance.

Performance optimization of caches. Several design techniques increase performance of the data cache.⁶ First, the cache behaves in a write-back or copy-back fashion, in which stores to cached locations do not show up on the

LD.L address, R5	F	D	A	C	W*			
ADDU R6, R7, R8		F	D	E	W			
BR (to target)			F	D	I	—		
ADDU R5, R5, R5				F	D	E	W	
SUBU (target)					F	D	E	W
Clock cycle	0	1	2	3	4	5	6	7

(a)

LD.L address, R5	F	D	A	C	W			
ADDU R5, R5, R5		F	D	—	E	W		
BR (to target)			F	—	D	I	W	
ADDU R6, R7, R8				—	F	D	E	W
SUBU (target)						F	D	E
Clock cycle	0	1	2	3	4	5	6	7

(b)

LD.L address, R5	F	D	A	C	ADS	—	RDY	W*
ADDU R6, R7, R8		F	D	E	W			
SUBU R9, R10, R11			F	D	E	W		
ADDU R12, R13, R14				F	D	E	W	
ADDU R15, R16, R17					F	D	E	W
SUBU R5, R6, R7						F	D	E
Clock cycle	0	1	2	3	4	5	6	7

(c)

ADS External bus address strobe
C Cache access
RDY External bus ready signal

Figure 10. Load pipeline: Instruction after load does not use loaded data (a); instruction after load with loaded data (b); and load instruction with data-cache miss (c). W* writes occur in a later clock cycle when the register file is free.

external bus. Modified data moves off the chip only when the cache line containing it must be replaced by a more recently requested line. If the alternative (a write-through cache) were used, the external bus would degrade performance in operations on large data arrays. Such operations include graphics transforms, matrix multiplies, fast Fourier transforms, and other digital signal processing functions.

In multiprocessors, write-back caches have a consistency problem when one CPU modifies data in its cache, unbeknownst to other CPUs. A write-through approach always updates main memory and other processors' caches. The write-back approach leaves cache consistency management to software, or to "bus snooping" hardware. The i860 XR CPU does not contain snooping hardware. Thus, i860 XR multiprocessor cache consistency requires software to avoid caching shared data, but the write-back approach improves bus traffic and performance for uniprocessors. The second-genera-

tion i860 XP CPU executes hardware snooping.

When write-backs do occur, they may require only two bus cycles. Since each cache line contains 32 bytes, writing an entire line would take four transfers on the 8-byte bus. However, the processor writes only half a line, when only half has been modified. It does so by keeping two dirty-status bits per line.

The on-chip cache makes load data available in the second instruction after the load, with only one delay slot. When the data is used in the very next instruction after the load, hardware makes the instruction wait for the data, as shown in Figure 10b, (unlike the Mips R3000, which depends on the compiler or programmer to avoid using the data in that next instruction).⁷ Figure 10 shows the load pipeline, which can be abbreviated FDACW (fetch, decode, address, cache access, write). Floating-point loads, while using the same FDACW pipeline, face two delay slots because the floating-point data is not "bypassed" from the cache directly to the execution units. The large number of floating-point data paths contributed to this decision.

The difference in length between the load pipeline (5 stages) and arithmetic execution (4 stages) can yield conflicts in use of the one integer-register file write port. For example, in Figure 10a the LD and ADDU instructions both need to write results in clock cycle 4. This conflict gets resolved without performance loss by delaying the load-write until a later time when no other instruction needs the write port. That time occurs in cycle 5 in the figure, a branch's fourth stage. The processor behaves as if the load write occurred at the normal time by feeding the load-data input register to any later instructions referencing it.

Accesses to the cache by the load instruction LD are fully scoreboarded: an internal state machine saves the destination register tag and checks that subsequent instructions do not use the same register. Scoreboarding allows execution to continue until the destination is referenced by a later instruction (Figure 10c). However, floating-point load (FLD) cache misses are not scoreboarded, because the large number of paths possible to incoming data in the floating-point registers made a floating-point scoreboard too costly. Thus a cache miss on FLD freezes the CPU until the first datum is available.

Stores are posted into either of two on-chip write buffers, allowing execution to continue despite write cache misses. The two write buffers, at 16 bytes each, can hold a modified line for a write-back that results from a replacement. They allow the line fill of requested data to occur without waiting for the write-back of the line it displaces. Furthermore, all cache line fills wrap around, so that the first instruction or

```

ld.l iterations, counter_reg
fld.d array_pointer++, freg      //get datum
loop:
  adds -1, counter_reg, counter_reg //decrement counter
  fadd.ss freg, sum, sum
  bnc.t loop                      //Delayed branch
  fld.d array_pointer++, freg      //Delay-slot instruction
(a)

ld.l iterations, counter_reg
pfld.d array_pointer++, f0        //prime the 3-stage pipeline
pfld.d array_pointer++, f0        //
pfld.d array_pointer++, f0        //
pfld.d array_pointer++, freg      //get first datum into freg
loop:
  adds -1, counter_reg, counter_reg
  fadd.ss freg, sum, sum
  bnc.t loop
  pfld.d array_pointer++, freg
(b)

```

Figure 11. Using the PFLD instruction: Code to sum a floating-point array (a) and similar code using PFLD (b).

datum the processor sees on a miss is the needed one. The other three transfers of the line occur in the background as the processor resumes execution. After they complete, an accompanying write-back can occur in the background.

During instruction cache line fills, the CPU can continue executing instructions as they are fetched, rather than waiting until the end of the line transfer. Because the 8-byte external bus transfers two instructions every two clock cycles, the CPU can copy the input register into the instruction register for execution.

To ensure speedy data cache flushes, designers implemented a FLUSH instruction. It empties the cache more than twice as quickly as the replacement operations described earlier. FLUSH causes modified data to be copied to the external bus and overwrites the tag directory with an innocuous value. While a normal cache-missing load instruction also would cause a dirty data dump, it would waste four bus cycles, bringing a new line into the cache. The FLUSH instruction must be executed once for each of the 256 cache lines, taking 50 μ s total at 40 MHz in the worst case when the entire cache is dirty but only 25 μ s when no lines are dirty.

Bypassing the cache. To allow the programmer to control which data gets into the cache, the processor provides a novel instruction, PFLD (pipelined floating-point load). The processor does not cache items fetched using PFLD. It intentionally bypasses the data cache to avoid thrashing, or displacement of still-needed cache data by newly fetched data.

Programmers can also make data noncacheable via the page-table entries or by external hardware decoding addresses to deactivate the cache enable (KEN#) pin, but these do not offer the optimizations found in PFLD.

The pipelined part of PFLD allows it to tolerate the increased latency of external memory compared to on-chip cache hits. The PFLD instruction returns data from the address generated by the third previous PFLD. This behavior enables the i860 processor to maintain the full bus bandwidth while accessing large data sets that reside in external memory, even if the latency of the memory is six clock cycles or greater. That is, the semantics of the instruction allow the program to issue the data request to memory many clocks before the resulting data is demanded. PFLD works similarly to the pipelined add PFADD and multiply PFMUL instructions. The destination register receives data from the end of a three-stage pipeline, rather than the result from the operation initiated using the two source registers specified in the instruction.

PFLD uses an on-chip, three-stage data FIFO (first-in, first-out buffer) situated between the data pins and the internal data bus. The i860 uses the FIFO only for fast external memory systems, which return data for PFLD requests before the subsequent PFLD instructions (which use the data) are encountered. Here "fast" is a relative term, as the sparse occurrence of PFLD in most programs makes any memory fast when compared to the long interval between PFLDs. Data flows directly from the pins to the destination register when the FIFO is empty, avoiding any extra delay. Thus the three stages of the PFLD pipe can occur either inside or outside the chip (as in Figure 7) or a combination of both.

Using a normal load for vector operations can result in cache thrashing when an entire cache line is fetched to reference one word of data one time only. In this case, the data cache becomes a liability demanding bus bandwidth unnecessarily. PFLD excels at processing large amounts of data from a slow memory system and allows mixing of fetches from the internal cache with external fetches. Furthermore, the chip maintains coherence for PFLD by checking the cache for the data requested, just in case the program has already loaded the data into cache using normal LD or FLD instructions.

PFLD turns out to be easy to use in programs. It is handy for loops that access data arrays, and for memory-block moves common in graphics and operating-system code. The data does not have to be floating point, and programs can effectively use PFLD on integers and pixel data. In fact, the "floating" in PFLD merely refers to the destination, the floating-point register file. To convert normal loads to pipelined loads, the

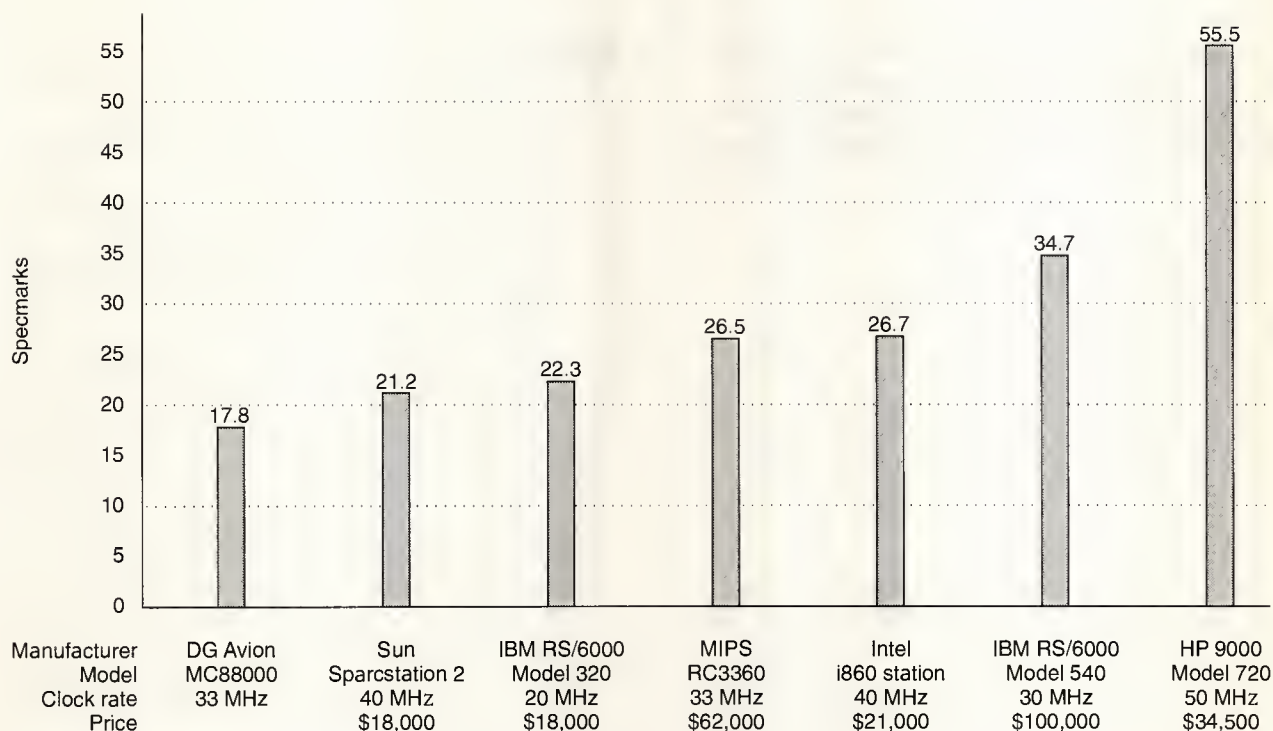


Figure 12. Specmarks and approximate prices for workstations (March 1991).

programmer merely includes three dummy PFLD instructions using the first three data addresses before the beginning of the loop (Figure 11). The internal FIFO takes care of the rest.⁸

Performance measurements

Good performance from a computer system depends on an efficient instruction set; its compilers, software libraries, and operating system; and a fast hardware implementation. However, evaluations should include dollar costs to have real meaning, as a Cray supercomputer outruns any workstation but is not cost-effective for many applications.

Of course, benchmarks may not match the real-world experience of the computer builder or buyer. But, given that every buyer cannot test all machines on a particular program suite, standardized benchmarks have become necessary input to buying decisions. The well-regarded SPEC benchmark from the System Performance Evaluation Cooperative⁹ uses 10 programs to measure CPU speed. With four C and six Fortran programs, it covers a wide range of scenarios. The Fortran programs use floating-point, making them appropriate for engineering users. SPEC does not test I/O performance extensively, and compiler quality can make or break SPEC performance. Figure 12 shows Specmarks for an i860

processor system under Unix Version 4.0 and other similarly priced RISC workstations, each at the maximum clock rate currently offered. Specmarks measure the speedup over a VAX 11/780, so higher Specmarks are faster.

Most RISC processors at a given clock frequency (33-50 MHz are maximum rates at the time of this writing) score similar SPEC results, because their instruction sets, caches, memory systems, and compilers are similar. However, Mips, Hewlett Packard, and IBM RS/6000 machines excel because of excellent compilers and RS/6000 superscalar execution,¹⁰ and the i860 excels because of fast floating-point operations.

Smaller benchmarks like the Whetstone and Dhrystone fit into the on-chip cache and exploit the 16-bytes-per-clock bandwidth of the cache for text-move operations. Likewise, the fast Fourier transform, which is ubiquitous in digital signal processing, proves extremely fast on the i860 CPU, making it popular in signal processing and graphics applications.¹¹ For the FFT, the i860 achieves nearly two instructions each clock cycle and 52 Mflops at 40 MHz (Table 1).¹²

Next-generation i860 CPU

The i860 XP microprocessor uses more than 2.5 million transistors in a submicrometer process to enhance perfor-

Table 1. FFT performance for the fastest chips available in 1990.¹² FFT benchmark is 1,024-point, single-precision, complex-input.

CPU	Clock rate (MHz)	FFT time (ms)
Intel i860	40	0.74
TI TMS320C30	33	3.4
AT&T WEDSP32C	50	2.8
Motorola 96002	33	1.13

A 1,024-point fast Fourier transform equals approximately 16,000 multiply and 29,000 add/subtract operations.

mance with new features and 32 Kbytes of on chip caches. It is i860 software binary-compatible. With additional registers, larger caches, a burst bus, and a 50-MHz clock rate, the CPU doubles the performance of the original i860 XR processor. To allow system performance to grow faster than clock rates, it also implements multiprocessor cache consistency and synchronization primitives. Intel introduced the i860 XP and accompanying second-level cache chips in June 1991.

AS A NEWER ENTRY IN THE MICROPROCESSOR RACE, the i860 architecture incorporates the state of the art in performance-seeking ideas. Novelty in the i860 CPU include simultaneous floating-point operations similar to digital signal processing, a two-instruction-per-clock mode, fast floating-point pipelines, graphics instructions, and high-bandwidth registers and caches on-chip. These features make it one of the fastest single-chip processors available, popular especially with graphics and supercomputer builders.

By including these features, the i860 architecture will exploit future advances in compiler technology; an architecture that does not offer explicit parallelism opportunity can expect performance growth only through faster clock rates and caches. ■

Acknowledgments

I thank Les Kohn, Neal Margulis, Piyush Patel, and Michael Rhodehamel for insights into the i860 architecture.

References

1. T.S. Perry, "Intel's Secret Is Out," *IEEE Spectrum*, Vol. 26, No. 4, Apr. 1989, pp. 22-28.
2. N. Margulis, *i860 Microprocessor Architecture*, Osborne/McGraw-Hill, 1990, pp. 4-5.
3. G. Radin, "The 801 Minicomputer," *IBM J. Research and Development*, Vol. 27, No. 3, May 1983, pp. 237-246.
4. H. Sit et al., "An 80-Mflops Floating-Point Engine in the Intel i860 Processor," *Proc. IEEE ICCD*, 1989, IEEE Computer Society Press, Los Alamitos, Calif., pp. 374-379.
5. L. Kohn and N. Margulis, "Introducing the Intel i860 64-Bit Microprocessor," *IEEE Micro*, Aug. 1989, pp. 15-30.
6. P. Patel and D. Douglass, "Architectural Features of i860 Microprocessor RISC Core and On-Chip Caches," *Proc. IEEE ICCD*, 1989, pp. 385-390.
7. M. Slater, "Mips Previews 64-Bit R4000 Architecture," *Microprocessor Report*, Vol. 5, No. 2, Feb. 6, 1991, p. 8.
8. M. Rhodehamel, "The Bus Interface and Paging Units of the i860 Microprocessor," *Proc. IEEE ICCD*, 1989, pp. 380-384.
9. "i860 64-Bit Microprocessor Performance Brief," Intel Corp., Aug. 1990, order #240588-004, pp. 3-7.
10. H. Bakoglu et al., "The IBM RISC System/6000 Processor: Hardware Overview," *IBM J. Research and Development*, Vol. 34, No. 1, Jan. 1990, pp. 12-22.
11. J. Grimes et al., "The Intel i860 64-Bit Processor: A General-Purpose CPU with 3D Graphics Capabilities," *IEEE Computer Graphics and Applications*, July 1989, pp. 85-94.
12. M. Stauffer and M. Slater, "Dual-Bus Design and IEEE Floating-Point Distinguish 96002 from Competitors," *Microprocessor Report*, May 30, 1990, Vol. 4, No. 10, p. 12.



Mark Atkins is a senior applications engineer for Intel. His interests include CPUs and multiprocessing. Before joining Intel in 1988, he designed RISCs at IBM for six years. Atkins received BSEE and MSEE degrees from Purdue University.

Address questions concerning this article to the author at Intel Corp., SC4-40, 2625 Walsh Ave., Santa Clara, CA 95051; or on e-mail at matkins@smdvxl.intel.com.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 159

Medium 160

High 161

Software Report

continued from p. 35

which also involves a considerable amount of money. Experts predict that it will take at least another 10 years for Japan to catch up with American MPU technology.

Techno Japan is published monthly by Fuji Technology Press Ltd. in Tokyo.

MITI programs

MITI's Intelligent Production System (the IPS 10-year project, US\$1.1 billion) plans participation from DEC, Rockwell, United Technology, Boeing, General Motors, IBM, Kodak, and Texas Instruments. Feasibility studies started in February.

MITI plans to expand its multi-language machine translation system development project to include European languages. This is a six-year project (\$150 million) begun in 1987 with participation of the Chinese, Thai, Malaysian, and Indonesian governments. The project uses an intermediate language and scheduled its first seminar with US and European Community governments last July. MITI plans call for an electronic dictionary and a prototype by 1995 and translation centers established in Asian countries in 1993-1995.

MITI's Basic Technology Center will conduct research on network-based learning and develop a system for providing interactive learning environments to network users. Some new companies will be funded to accomplish this (about US\$30 million).

Software structuring program

MITI has upgraded the New Software Structure Model to the status of an international research project and set up two special research groups in the Japanese Information Processing Association, which began activities in October

1990. The purpose of this project is to develop a system in which software will be flexible enough to deal with errors and other changing external conditions. The project is an industry-university cooperative program. Overseas participants include Stanley Peters from Stanford University and Joseph Gogian from Oxford University.

Production systems

The company's industrial structure is changing into multiple levels. In particular, experts expect a shift into a tertiary sector centered around service to continue. Economic and social conditions are moving from quantitative expansion to qualitative improvements. Market needs will force manufacturing industries to change from large quantities of a few products to variable quantities of different products. Manpower and labor shortages have already appeared. We've seen a noticeable trend among science/engineering people to stay away from manufacturing. The country has an urgent need to streamline facilities using mechatronics (computers), and to shift from streamlining individual facilities to entire manufacturing systems. The focus of business must be shifted from "things" to information."

Specifications and quantities of products will become individualized. So, production systems should be able to provide products based on unique ideas. These systems must freely multiply or shrink and even change functions. In addition, quick delivery after order receipt is needed, as are distributed databases that are immediately updatable. Production systems with high fault-tolerance and automated functions (monitoring, malfunction diagnosis, restoration) are needed to carry out unmanned operations. Production systems must be flexible to minimize remodeling expenses and time, as well as the chances of the systems becoming obsolete as a result of product/model/schedule changes. Intelligent human interface support systems are

needed.

The most important tasks include product design systems (CAE/CAD/CAM) and production process design systems (computer assembly to manage CAD/CAM). The country has a high demand for advanced supervisory systems, though it rates automated restoration systems lower. Immediate goals are in mechanical systems for diagnostic support. Expected by 1995-1998, and rated higher by manufacturers than by users, are automating visual observations and other sensing tests. In addition, design operations need to be standardized. Plans call for development of a comprehensive CAD/CAM/CAE system by 2000-2005.

Production engineering know-how must be transformed into databases to bring about widespread use by 2000. Production management systems, though developed early, will be affected by the development of databases and expert systems. An automated assembly system should be capable of copying an engine by 1995-1998.

Intelligent robots face problems in building other robots and interfacing with peripherals. Developments in artificial intelligence, such as 3D recognition, and neural/fuzzy systems are expected by 1995-2000.

References

1. *Techno Japan*, Vol. 23, Nos. 8-9, Aug. and Sept. 1990.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 195 Medium 196 High 197

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Northwest Region: D. Rodney Brooks; Tel: (415) 905-0260; Fax: (415) 896-1512.

Eastern Region: Georgette Boone; Tel: (415) 905-0260; Fax: (415) 896-1512.

Southwest Region: Kenneth Smitley; Tel: (415) 905-0260; Fax: (415) 896-1512

Recruitment and Classified Advertising: D. Rodney Brooks; Tel: (415) 905-0260; Fax (415) 896-1512.

Director of Sales: Randall L. Stickrod, 544 Second St., Suite 200, San Francisco, CA 94107; Tel: (415) 905-0260; Fax: (415) 896-1512.

For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

IEEE MICRO, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

RS # Page #

Accusys, Inc.	43	40
Adobe Systems	85	42
Advanced Technologies Int'l	33	39
Allied Telesis Inc.	23	38
Applied Microsystems	86	42
Ariel Corp.	26	38
Aries Electronics, Inc.	36	40
AST Research Inc.	82	42
Bridgenorth Signal Processing Inc.	30	39
Burr-Brown Corp.	27	38
Cambridge Computer Corp.	17	37
Circuit Search	87	43
Cirrus Logic, Inc.	34	39
Cubit	90	43
Data Translation	28	39
Digital Communications Associates	16	37
Digital Vision	83	42
Dolch Computer Systems	14	37
Durham Technical Images	25	38
Electrim Corp.	42	40
Fotec Inc.	45, 46	41
Gespac	84	42
IBus PC Technologies	41	40
ICS Electronics	20	37
IEEE CS Press	—	34, 72
Inset Systems	88	43
Linear Technology Corp.	31, 37	39, 40
Martech	91	43
Maximum Strategy	13	37
Microtek International, Inc.	44	41
MNC International	39	40
Motorola	19, 32	37, 38
Myriad Solutions Ltd.	40	40
National Instruments	89, 92	43
NCR Corp.	12	36
NEC Technologies, Inc.	11	36
PC/M Corp.	24	38
Pere Line Data Systems	21	37
Practical Peripherals, Inc.	22	38
Qume Corp.	80	42
Specialix Inc.	38	40
Specialized Systems Consultants	93	43
Spectrum Signal Processing Inc.	29	39
Standard Microsystems Corp.	35	39
STN International	—	59
Sun Microsystems	10	36
Telex Communications	81	42
Tiara Computer Systems	18	37
Unisys Corp.	15	37



Have you heard about our...

**Technical Committee on
Microprocessors and Microcomputers**

*For information on this, or any of our more than
30 technical committees, contact:*

IEEE COMPUTER SOCIETY
Membership/Circulation Dept.
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1264
(714) 821-4010

1951 - 1991
40 YEARS OF SERVICE



IEEE COMPUTER SOCIETY

A member society of the
Institute of Electrical and Electronics Engineers, Inc.

THE FOLLOWING INFORMATION IS AVAILABLE:

Contact the Publications Office, to facilitate handling, please request by number.

- Membership application, student #203, others #202
- Periodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Comppmail electronic mail brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures #193
- Student scholarship information #192
- Volunteer leaders/staff directory #196
- IEEE senior member grade application #204

(requires ten years practice and significant performance in five of those ten)

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

PURPOSE

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

MEMBERSHIP

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and product reviews.

Periodicals. The society publishes six magazines and five research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

OMBUDSMAN

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

EXECUTIVE COMMITTEE

President: Duncan H. Lawrie*
University of Illinois
Dept. of Computer Science
1304 W. Springfield
Urbana, IL 61801
(217) 333-3373

President-Elect: Bruce D. Shriver*
Past President: Helen M. Wood*

VP, Standards: Paul L. Borrill (1st VP)*
VP, Press Activities: Barry W. Johnson (2nd VP)*
VP, Conferences and Tutorials: Laurel V. Kaleda†
VP, Education: Raymond E. Miller†
VP, Membership Activities: Ronald D. Williams†
VP, Publications: Ronald G. Hoelzeman†
VP, Technical Activities: Mario R. Barbacci*

Secretary: James H. Aylor*
Treasurer: Joseph Boykin†
Division V Director: Edward A. Parrish, Jr.†
Division VIII Director: Helen M. Wood*
Executive Director: T. Michael Elliott*

*voting member of the Board of Governors
†nonvoting member of the Board of Governors

BOARD OF GOVERNORS

Term Expiring 1991:

P. Bruce Berra, Michael Evangelist,
Ted Lewis, Raymond E. Miller, Earl E. Swartzlander, Jr.,
Joseph E. Urban, Thomas W. Williams

Term Expiring 1992:

James H. Aylor, Alicia I. Ellis, Tadao Ichikawa,
C.V. Ramamoorthy, Sallie V. Sheppard,
Harold Stone, Akihiko Yamada

Term Expiring 1993:

Fiorenza Albert-Howard, Jon T. Butler, Michael C. Mulder,
Yale N. Patt, Anneliese von Mayrhauser,
Benjamin W. Wah, Ronald Waxman

Next Board Meeting

November 1, 1991, 8:30 a.m.
Opryland Hotel, Nashville, TN

SENIOR STAFF

Executive Director: T. Michael Elliott
Publisher: H. True Seaborn
Director, Conferences and Tutorials: Anne Marie Kelly
Director, Finance and Administration: Tod S. Heisler
Director, Board and Administrative Services: Violet S. Doan

COMPUTER SOCIETY OFFICES

Headquarters Office

1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone (202) 371-0101
Fax: (202) 728-9614

Publications Office

10662 Los Vaqueros Cir.
PO Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information:
(714) 821-8380
Publication Orders: (800) 272-6657
Fax: (714) 821-4010

European Office

13, Ave. de L'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05

Asian Office

Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 3408-3118
Fax: B1 (3) 3408-3553



IEEE OFFICERS

President: Eric E. Sumner
President Elect: Merrill W. Buckley, Jr.
Past President: Carleton A. Bayless
Secretary: Hugh Rudnick
Treasurer: Theodore W. Hissey, Jr.

VP, Educational Activities: Richard S. Nichols
VP, Professional Activities: Michael J. Whitelaw
VP, Publication Activities: J. Thomas Cain
VP, Regional Activities: Robert T. H. Alden
VP, Technical Activities: Fernando Aldana



1992

Editorial Calendar

FEBRUARY

Updates on microprocessor/microcontroller developments

- Script-recognition systems
- Message-passing techniques
- High-speed, low-cost system communications
- Floating-point processors

Ad closing date: January 1

AUGUST

European industry

Recent developments in integrated circuit and microsystem technology from major European manufacturers

Ad closing date: July 1

APRIL

Hot Chips III

- This extremely popular issue presents the latest developments in microprocessor and chip technology used to construct high-performance workstations and systems as presented at the annual IEEE Computer Society-sponsored symposium
- Past contributors include: Intel, Motorola, Apple, Sun, IBM Research, Metaflow Technologies, Intergraph, ITT

Ad closing date: March 1

OCTOBER

ICs for HDTV

- Coordinated issue with *IEEE Computer Graphics and Applications* magazine
- The high-definition television field requires support from IC manufacturers to ensure its first successes
- What are governments doing to support HDTV?

Ad closing date: September 1

JUNE

Associative memories and processors

- Late-breaking developments in wide-word content-addressed memories (CAMs)
- Dynamic CAPP (parallel processor) architectures
- Fault-tolerant architectures for crucial control systems such as those in trains, space systems, etc.

Ad closing date: May 1

DECEMBER

Special Signal Processors

- Recent information from the vital area of digital signal processors
- News about other techniques for signal processing
- Mixed analog/digital processors solve a real need
- Neural networks process signals following methods used by the human brain

Ad closing date: November 1

IEEE Micro helps designers system integrators, and users of microprocessor and microcomputer systems explore the latest technologies to achieve business and research objectives.

Feature articles in *IEEE Micro* reflect original works relating to the design, performance, or application of microprocessors and microcomputers.

All manuscripts are subject to a peer-review process consistent with professional-level technical publications. *IEEE Micro* is a bimonthly publication of the IEEE Computer Society.

Advertising information: Contact Marian Tibayan, Advertising Department, IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax (714) 821-4010.

Articles may change. Please contact editor to confirm.

IEEE

MICRO

Chips, Systems, Software, and Applications

DECEMBER 1991

1 Here, There, Everywhere	4 China Doll	7 Georgia on My Mind
2 Girl from Ipanema	5 St. Louis Blues	8 Hotel California
3 Man of La Mancha	6 Blue Danube Waltz	9 Lullaby of Birdland



DATABASE MACHINES

*Companion Issue
to December 1991
Computer on
Heterogeneous
Distributed
Database Systems*

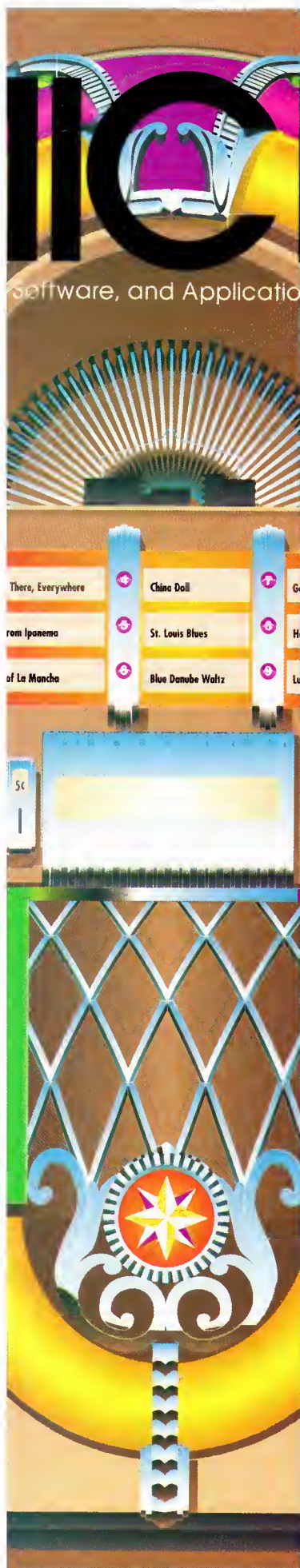
1961-1991



40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.



IEEE MICRO

Published by the IEEE Computer Society

December 1991

F E A T U R E S

6 Guest Editors' Introduction: Database Machines—Trends and Opportunities

M. Abdelguerfi and A.K. Sood

8 VLSI Accelerators for Large Database Systems

*Kuo Chu Lee, Takako Matoba Hickey,
Victor W. Mak, and Gary E. Herman*
Resolving the problem of slow response
times in a cost-effective manner

22 An Associative Accelerator for Large Databases

Pascal Faudemay and Mongia Mhiri
Implementing relational operations at
speeds adaptable to advanced micropro-
cessors

35 A Fine-Grain Architecture for Relational Database Aggregation Operations

M. Abdelguerfi and A.K. Sood
Designing and simulating a prototype
four-input unit for fabrication using
discrete components

44 A Parallel, Scalable, Microprocessor- Based Database Computer for Performance Gains and Capacity Growth

David K. Hsiao

Using a variable number of processors to
produce an experimental computer

61 Rinda: A Relational Database Processor with Hardware Specialized for Searching and Sorting

*Ushio Inoue, Tetsuji Satoh, Haruo Hayami,
Hideaki Takeda, Toshio Nakamura, and
Hideki Fukuoka*

Reducing a host computer's CPU and I/O
times with specialized hardware

71 Special Feature Annual Index—Volume 11

Cover design: Alexander Torres

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; IEEE Computer Society Headquarters, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription: \$21 in addition to IEEE Computer Society or any other IEEE society member dues; \$38 for members of other technical organizations. This journal is also available in microfiche form.

Postmaster: Send address changes and undelivered copies to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Second-class postage is paid at New York, NY, and at additional mailing offices.

Copyright and reprint permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US Copyright Law for private use of patrons those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress St., Salem, MA 01970. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to Permissions Editor, *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Copyright © 1991 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved.

DEPARTMENTS

- 2 Letters/Mailbag**
On an open architecture
- 4 Micro News**
PLA copyright infringement
- 78 Micro Law**
Database system copyrights
- 80 On the Edge**
SBus: An open architecture
- 84 Micro Standards**
Computing interconnections
- 86 New Products**
Workstations; boards and cards;
communications; chips and
components

CS membership application, p. 21; Reader Interest/Service/Subscription cards, p. 64A; Advertiser/Product Index, cover 3; CS information page, cover 4

IEEE Computer Society
PO Box 3014, Los Alamitos, CA 90720-1264
(714) 821-8380

Editorial: Unless otherwise stated, bylined articles and descriptions of products and services reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society. Send editorial correspondence to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. All submissions are subject to editing for style, clarity, and space considerations.

EDITOR-IN-CHIEF

Dante Del Corso
*Politecnico di Torino**

ASSOCIATE EDITOR-IN-CHIEF

Ashis Khan
*Mips Computer Systems, Inc.***

EDITORIAL BOARD

John Crawford
Intel Corporation

K.E. Grosspietsch
GMD

Joe Hootman
University of North Dakota

David K. Kahaner
National Institute of Standards and Technology

Hubert D. Kirmann
Asea Brown Boveri Research Center

Priscilla Lu
AT&T

Richard Mateosian

Nadine E. Miner
Sandia National Laboratories

Ken Sakamura
University of Tokyo

John L. Schmalzel
University of Texas at San Antonio

Michael Slater
Microprocessor Report

John W. Steadman
University of Wyoming

Richard H. Stern

Philip Treleaven
University College London

Carl Warren
McDonnell Douglas Space Systems Co.

Maurice Yunik
University of Manitoba

MAGAZINE ADVISORY COMMITTEE

James J. Farrell, III (chair)
Fiorenza Albert-Howard

Valdis Berzins
Jon T. Butler

B. Chandrasekaran
Carl Chang
Manuel d'Abreu
Dante Del Corso
Gerald Engel
Michael Evangelist
Sushil Jajodia
H.T. Seaborn
Pradij K. Srimani
Peter R. Wilson

STAFF

Marie English
Managing Editor

David Sims
Assistant Editor

H.T. Seaborn
Publisher

Marilyn Potes
Editorial Director

Douglas Combs
Assistant Publisher

Pat Paulsen
Assistant to the Publisher

Jay Simpson
Art Director

Joseph Daigle
Production

Christina Champion
Membership/Circulation Manager

Heidi Rex,
Marian Tibayan
Advertising Coordinators

PUBLICATIONS BOARD

Ronald G. Hoelzeinan (chair)

James Aylor
Victor R. Basili
Richard Burke
Jon T. Butler
J.T. Cain

B. Chandrasekaran
Carl Chang
Manuel d'Abreu
Dante Del Corso
Gerald Engel
Michael Evangelist
James J. Farrell, III
Tse-yun Feng
Anil K. Jain
Ted Lewis
Ray Miller
Michael C. Mulder
C.V. Ramamoorthy
H.T. Seaborn
Sallie Sheppard
Harold Stone
Earl E. Swartzlander
Peter R. Wilson

* Submit six copies of all articles and special-issue proposals to Dante Del Corso,
Dipartimento di Elettronica, Politecnico di Torino,
C.so Duca degli Abruzzi, 24, 10129 Torino, Italy; phone +39 11 564 4044;
Compmail: d.delcorso; Internet: delcorso@polito.it; Bitnet: delcorso@itopoli
or

** Ashis Khan. Mips Computer Systems, Inc., 950 DeGuigne Drive, Sunnyvale, CA 94086;
Internet: ashis@mips.com.



On an open architecture

To the Editor:

In the October 1991 issue I particularly liked "A RISC Processor for Embedded Applications Within an ASIC," "Performance and the i860," Micro News, Micro Law, and Software Report.

One criticism: In "On the Edge" about IEEE Std P1754, the P1754 architecture is compared to several commercially available microprocessors in Table 1. This table is preposterously misleading. It is clearly biased toward the P1754 architecture. For example, it claims that there are "many" performance grades of P1754 but "few" of the 68000 architecture, and states that those are "mainly different clock speeds."

The author describes different performance grades as differing in cache implementation, external bus structure, basic technology, etc. Without even looking up references, I can cite the following variants on the 68000 architecture which differ in these respects:

68008—8-bit data bus, 20-bit address bus;

68000—16-bit data bus, 24-bit address bus;

68HC000—CMOS;

68020—256-byte instruction cache, extra address modes, 32-bit address and data buses, dynamic data bus sizing;

68030—separate 256-byte instruction and data cache, MMU;

68040—4K separate instruction and data cache, MMU, FPU, bus snoops;

68331—added telecom features, also DMA and other features;

68340—essentially 68332 but with simple counter/timers; and

68EC040—68040 without FPU, MMU.

Table 1 also claims that the 68000 architecture has only one implementer. I believe the Phillips/

Signetics SCC68070 "highly integrated microprocessor" is a contradicting example.

I hope that *Micro* will be more successful in the future at ensuring that its content is accurate and not overtly biased.

Joseph M. Schachner
Spring Valley, NY

Reply:

Thank you for your comments regarding the P1754 article I wrote. I agree that my table may have been a bit misleading.

The point I was trying to make was not that other architectures don't have variations but that they have very limited variations based on available devices. The implementation of P1754 is open. P1754 may be implemented as a low-performance, low-cost device or as a high-performance superscaler.

Even so, I believe the 68K family itself is large, and each architecture variation (68000, 020, 030, 040, ...) is unique. P1754 is based on the Sparc family of RISC processors. This family also has different variations: Version 7, P1754, Version 8. Picking one variation of the Sparc family that has been made a standard and comparing it to one member of the 68K family would be a more appropriate comparison.

You see, the concept of P1754 is very different from the 68K family (for that matter, from most microprocessor architectures). P1754 defines an Instruction Set Architecture (ISA) and not a device. And as such, the ISA is mostly flexible in performance grades and implementations (i.e., cache, external bus structure, basic technology).

Again, it was not my intention to insinuate that other architectures (specifically 68K, 88K, i860) are not as good or less powerful but to point out how different the definition of P1754 is.

Rudolf Usselmann

In the mailbag

(LK: liked, DLK: disliked, LTS: like to see)

Many readers continue to write about the split articles. They get the answer in this issue. From this experiment, we learned that readers do care about *Micro*. Thanks, and please continue to write and to care!—D.D.C.

December 1990

LTS: I suggest you send all the subscribers the index that appears in the issue of December of each year on a floppy disk.—J.L.B., Montevideo, Uruguay (Your suggestion sounds reasonable to me. If it proves cost-effective, I would propose it.—D.D.C.)

LK: Your articles on the ESPRIT project; what ... about the Japanese project for the fifth-generation com-

puter? LTS: Articles (cluster of articles) on other projects like those under the ESPRIT umbrella (even from Japan....)—B.M., Timisoara, Romania

February 1991

LK: "The Drive to the Year 2000"—F.P.B, Laurel, MD

DLK: The new practice of breaking up the articles into widely separated fragments is very poor.—S.L.P., Darien, CT

April 1991

LK: Analysis of multicomputer/multiprocessor systems; DLK: jumping across articles; LTS: the architecture of MIPS R4000, FPU MC68882.—K.V., Bangkok, Thailand

DLK: Your new magazine format is terrible!—K.R., North Amherst, MA

LK: The article on communication latency; it was well presented; LTS:

more articles analyzing the architecture/ performances of various machines.—A.S.M., Secunderabad, India

LK: "The end of the 386 monopoly"—the reportage style was clear and interesting; LTS: in the Micro Law column, an article on patent law changes with respect to micro applications.—D.S., Philip, Australia

LK: Micro News, Micro Review, Micro View, Micro Law, New Products. I wait for "Hot Chips." DLK: Hated: split articles! Yes, I have read the editor's reply on p. 2. Personally, I would prefer no color page at all to a split, (sliced and mixed) magazine. I'm sorry to say that.—T.P., Warsaw, Poland

June 1991

LK: Micro News... W.M.S., Atlanta, GA

NEW Conference Proceedings from IEEE Computer Society Press

12TH REAL-TIME SYSTEMS SYMPOSIUM

The proceedings covers state-of-the-art research and key developments in real-time computing and discusses issues such as hard real-time communications, priority scheduling, object-oriented modeling, optimization, real-time databases, requirements specification models, and adaptive real-time systems. Its articles discuss new research on improvements in the construction of real-time systems and to enhance the growth in real-time computing R&D.

320 PAGES. DECEMBER 1991. SOFTBOUND. ISBN 0-8186-2450-7.
CATALOG NO. 2450 \$70.00 MEMBERS \$35.00

8TH TRON PROJECT SYMPOSIUM

The TRON Project Symposium features four sessions providing detailed explanations of ITRON (industrial), BTRON (business), CTRON (central and communications), and TRON-specification VLSI CPU. It includes 19 articles covering implementation and application issues, and explores new specification proposals for next generation products based on the TRON architecture. In addition, it investigates the current efforts by researchers to develop a standardized and open architecture covering the area from operating system to VLSI chips.

264 PAGES. NOVEMBER 1991. SOFTBOUND. ISBN 0-8186-2475-2.
CATALOG NO. 2475 \$60.00 MEMBERS \$30.00

VLSI DESIGN 1992 — 5TH INTERNATIONAL CONFERENCE ON VLSI DESIGN

The proceedings of *VLSI Design '92* provides an extensive discussion of the latest advances in technology and explores recent technical opportunities in electronic design automation, and consists of over 80 papers on advanced test methods, physical design, design and fabrication, logic and high-level synthesis, VLSI tools and technology, testing, VLSI architectures, control and data path synthesis, signal processing applications, design for testability, layout, verification, industrial applications, and design verification.

400 PAGES. JANUARY 1992. SOFTBOUND. ISBN 0-8186-2465-5.
CATALOG NO. 2465 \$90.00 MEMBERS \$45.00

1ST INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED INFORMATION SYSTEMS

This book includes over 40 articles on current applications, and is divided into 12 sections that investigate information systems technologies such as disk replication and arrays, object-oriented systems, joins, parallel logic, transitive closure and synopsis, retrieval, distributed systems, architectures/parallel processing, file systems, shared memory systems, and shared nothing systems.

320 PAGES. DECEMBER 1991. SOFTBOUND. ISBN 0-8186-2295-4.
CATALOG NO. 2295 \$64.00 MEMBERS \$32.00



To order call toll-free 1-800-CS-BOOKS





Send information for inclusion in *Micro News* one month before cover date to Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

Intel sues AMD for PLA copyright infringement

Richard H. Stern, Contributing Editor

The first significant copyright infringement suit based on a hardware device, other than code stored in a ROM chip (see box on copyright cases), was filed in federal court in Austin, Texas, on October 9, 1991. Intel Corp. sued Advanced Micro Devices for infringing on a copyright in what Intel identified as a "computer program stored in PLA [programmable logic array] of 80386 microprocessor" chip.

(In another part of Intel's complaint, it accused AMD of infringing on an Intel copyright in the microcode of the same chip. Intel and AMD are also engaged in patent, copyright, and mask work infringement litigation in federal court in San Jose,

California, over 80286 and 80287 chips. In addition, they have been engaged in an arbitration proceeding, with related litigation, over their chip technology interchange and second-sourcing agreements.)

Control program. Intel's complaint against AMD states that the 386 PLA contains the "control program" for the 386 DX and SX chips. Intel did not describe the function of the so-called control program (the company's designation; it is not a recognized term of art), but it appears that this PLA acts as a decoder for macro instructions. That is, an assembly-code instruction, such as "Move contents of register A to register B," may be executed by five or more steps represented in microcode. The microcode ROM of a microprocessor such as the 386 stores the instructions for these steps, as well as the steps for

PLA and ROM copyright cases

In 1987 Alloy Computer Products brought two suits in federal court in Los Angeles, charging copyright infringement of an alleged computer program stored in a PLA device.^{1,2} Apparently, these cases never came to trial, perhaps because the defendants surrendered without contesting the claim. No reported precedent on copyright in PLAs has emerged from these suits.

Federal case law has established that operating system software stored in ROM is subject to copyright protection.³ This principle apparently extends to microcode sorted in ROMs in microprocessors. However, the one decision on that issue concluded that the particular microcode involved (8086/8088) was

not infringed because the accused microcode (V20/V30) was too different to be found "substantially similar" (the test for copyright infringement).⁴

References

1. Alloy Computer Prods. v. Ultratek Corp., No. 87-6993 (C.D. Cal. filed Oct. 20, 1987).
2. Alloy Computer Prods. v. Asadi, No. 87-1285 (C.D. Cal. filed Mar. 23, 1987).
3. Apple Computer, Inc. v. Franklin Computer Corp., 714 F. 2d 1240, (3d Cir., 1983); cert. dismissed by stip., 464 U.S. 1033 (1984).
4. NEC Corp., v. Intel Corp., 10 USPQ 2d 1177 (N.D. Cal. 1989).

other instructions.

What Intel describes in its complaint as the control program may therefore be a symbolic notation for a decoder PLA that receives as input the object code for an expression such as, "Move contents of register A to register B." The decoder PLA then provides as output a starting address in the microcode ROM for the relevant five or more steps. Such a decoder would be, essentially, an array of gates (such as And and Or gates) corresponding to a set of Boolean equations for transforming instructions of the sort described (placed into object code format) into appropriate addresses of the microcode ROM.

PLA contents are usually specified symbolically. This symbolic representation is presented to the device that creates the hardware representation. Since the symbolic representation describes the relationship between inputs and outputs of the PLA, essentially Intel is asserting a copyright in what the PLA does, rather than the specific way it does it. Query: Can this be meaningfully distinguished from a copyright in a source code, which is copyrightable?

There are no close legal precedents on whether information in this form is a "computer program," as that term is defined in the Copyright Act. (Section 101 of the Copyright Act defines a computer program as a set of statements or instructions used to bring about a result in a computer.)

There are also no close legal precedents on whether making and selling a physical device—for example, a 386 clone—containing a substantially similar array of gates infringes on a copyright in the information describing the array. Such copyrights are registered with the Copyright Office by filing the information in some sort of printout (perhaps the Boolean equations, or an object code printout in binary or Hex of the microcode addresses tabulated against the object code inputs). Query: Is the physical device a legally protected "copy" of what was registered?

Semiconductor Chip Protection Act

The US Copyright Office's refusal to issue copyright registrations for semiconductor chip layouts led to passage of the Semiconductor Chip Protection Act of 1984. This law covers layouts of unprogrammed PROMs—programmable ROMs—but apparently does not cover layouts, or mask works, of programmed PROMs blown from such unprogrammed PROMs. It also covers layouts of both personalized and unpersonalized gate arrays. Probably, it would cover the layout of the PLA involved in the *Intel v. AMD* case. But a functionally similar PLA with a different layout from the registered layout would not be covered by such a registration.

Did the Copyright Office know?

Intel attached the copyright registration certificate (No. TX 3 121 803) for its PLA to its complaint. Such certificates include a "correspondence" box that the Copyright Office checkmarks if correspondence occurred between the office and the registrant. The box on this certificate is not checked.

That omission may suggest that the Copyright Office rubber-stamped the application without realizing that a PLA was involved (assuming that the Copyright Office even knows what a PLA is), and without realizing that the computer program in question was perhaps imaginatively so designated. One might expect the Copyright Office, if it knew that a gate array was being claimed as an information storage device, would ask for an explanation.

Further, considering the Copyright Office's history of resistance to copyrights on hardware, (see box on Semiconductor Chip Protection Act) one might expect it to issue gratuitous

advice to Intel that the copyright did not extend to physical devices built in accordance with or embodying the information. It is common practice for the Copyright Office to do so, and to place a record of the advice, in the form of a letter, in the official file.

None of the foregoing suggests that an explanation about the PLA, if Intel had given one to the Copyright Office in response to such a demand, would necessarily have been unsatisfactory. There is a great deal of isomorphism between 1s and 0s in a ROM and the equivalent in a PLA. One device Ands sets of Or gates, while the other device Ors sets of And gates. It may well be that analogous assembly codes can be created and described for each device.

Both such codes (if they are accepted as being codes) might be compiled or assembled into 1s and 0s (connections and open circuits) in analogous ways. But one would have expected the Copyright Office to ask for such an explanation if it understood what was going on in the case of this copyright application. Perhaps AMD's eventual response to Intel's claims will shed more light on these issues.

If PLA programs are held protectable by copyright, that will have startling and important implications for designers of electronic circuitry. (For a further discussion of the issue, see *Micro Law*, "Field-programmable logic devices—Are they hardware or software? Can their programmed configurations be protected against copying?" in *IEEE Micro*, Vol. 6, No. 5, Oct. 1986, pp. 61-62 and 78.)

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 198 Medium 199 High 200



Guest Editors' Introduction

Database Machines: Trends and Opportunities

M. Abdelguerfi

University of New
Orleans

A.K. Sood

George Mason
University

As could be expected, demands of the user community and developments in computer hardware and software have affected database technology. Although, traditionally, database practitioners worked with well-structured or semistructured databases, the development of new and reliable sensing techniques added impetus to research of unstructured databases such as those containing images.

These trends focused interest on unconventional database applications, despite their very large data volumes. For example, a 150-bed hospital generates 2 Gbytes of image data each day, and the US National Aeronautics and Space Administration's planned Earth Observing System program could generate several Gbytes of data every day. The archiving, retrieval, and management of such large databases constitute a complex and difficult task. Recent trends in database technology suggest that database computers can best undertake the efficient management of large databases. That is, specialized hardware and software configurations aimed primarily at handling large databases and answering complex queries become the best solution.

In general, most of the data modeling effort focuses on three widely used, structured database models: relational, hierarchy, and network. Other, recently proposed object-oriented data models may be suitable for modeling databases that include structured and unstructured data. Since the theoretical underpinnings of the relational model have been well defined, the relational model became the focus of most of the commercial and research efforts in database machines. Some additional effort in the development of hardware solutions for semistructured databases like text retrieval also surfaced. The articles in this issue reflect these database machine trends. Four of them discuss issues related to relational databases,

and another delves into text retrieval.

The primary motivation of the database machine approach is to increase the performance of updating operations and the query processing of databases. One approach taken to achieve this objective employs parallel architectures. The designer must choose the appropriate approach for data distribution and storage, interprocessor communications, and computational capability of each node. We can divide the parallel architectures studied in the context of database processing into three groups: shared memory, shared disk, and "shared nothing."

The current trend in the design of database computers is toward the increasingly popular shared-nothing¹ message-passing architectures. Designers usually prefer these architectures over shared-memory and shared-disk architectures because they permit a high degree of scalability. The MDBS database computer described by Hsiao in this issue belongs to this category.

Database engines^{2,3} represent another example of shared-nothing database systems. The high-performance and fault-tolerant database engines generally support a server/client architecture. A number of new database engine products have already hit the market, and more are currently under development. Teradata's DBC/1012 uses 80386 processors (up to 1,024). Because processors can be added as needed to increase both CPU and input/output capabilities, this database engine can handle very large databases.⁴ White Cross offers a database engine based on the Inmac transputer. The hardware and software structure in this database engine permits a maximum of 100,000 MIPS of processing power.⁵

The articles in this special issue on database computers fit into two main classes. The first class centers on hardware accelerators for database computers, and the second class presents two different database computers. The proposed hard-

Companion issue to December 1991 *Computer on Heterogeneous Distributed Database Systems*

ware accelerators concentrate on the efficient implementation of specific database tasks. The articles on database computers deal with the overall architecture, performance, and data distribution issues; they also discuss specific hardware accelerators incorporated in the systems.


The first three articles examine the design and evaluation of special-purpose accelerators for database computers. Lee et al. present two VLSI accelerators for formatted and unformatted databases. The database tasks performed by the accelerators include operations such as associative search, aggregation, and string search.

Faudemay and Mhiri present an associative accelerator whose speedup ratio is independent of the database size. The accelerator implements relational database operations and sorting and aggregate functions.

Our article describes the design and simulation of a bit-serial accelerator for statistical aggregation operations. A number of bit-serial processing elements connected according to the odd-even network topology make up the accelerator. The proposed unit achieves a high degree of pipelining and parallelism.

The remaining two articles describe two database computers and emphasize both architectural and performance considerations. Hsiao presents an experimental database computer with a variable number of database processors known as the Multiback-end Database Supercomputer (MDBS). Each micro-processor-based processor has a private database store, consisting of a small Winchester drive for paging and metadata and a large drive for the database.

Finally, Inoue et al. describe a relational database processor with specialized hardware for searching and sorting known as Rinda. Rinda is composed of two hardware accelerators, the CSP (Content Search Processor) and ROP (Relational Operation Accelerating Processor). The CSP searches rows of data on a disk storage and transfers the selected rows to the main memory. The ROP subsequently sorts the row transfers to the main memory.

We thank the reviewers who helped us referee the submitted papers and the *IEEE Micro* editorial staff. This special issue would not have appeared without their assistance. 

References

1. D.J. DeWitt et al., "A Single User Evaluation of the GAMMA Database Machine," in *Database Machines and Knowledge Base Machines*, M. Kitsuregawa and H. Tanaka, eds., Kluwer Academic Publishers, Boston, 1988; also in *Proc. Fifth Int'l Workshop Database Machines*, Oct. 1987, pp. 370-386.
2. V.A. Topkar, O. Frieder, and A.K. Sood, "Duplicate Removal on Hypercube Engines: An Experimental Analysis," *Parallel Computing*, North Holland (to be published).

3. O. Frieder, V.A. Topkar, R. Karne, and A.K. Sood, "Experimentation with Hypercube Database Engines," *IEEE Micro* (to be published in Feb. 1992).
4. Teradata Corp., *DBC/1012: Data Base Computer System—Introduction*, Boulder, Colo., 1986.
5. M. Butler and R. Bloor, "Client-Server Engines," *DBMS*, June 1991, pp. 17-18.



M. Abdelguerfi, an associate professor of computer science at the University of New Orleans, actively participates in research on database and knowledge-base machines, information retrieval, database management, design and analysis of parallel architectures, and VLSI architectures for nonnumeric computations.

Abdelguerfi received the Diploma in electrical engineering from the National Polytechnic School of Algiers, Algeria, and an MS and PhD in computer engineering from Wayne State University, Detroit. He is a member of the IEEE, the IEEE Computer and Circuits and Systems societies, the Association of Computing Machinery, Eta Kappa Nu, and Tau Beta Pi.



Arun K. Sood is a professor of computer science at George Mason University. His research interests include image analysis, signal processing, parallel and distributed processing, database machines, performance modeling, and optimization. His research efforts have resulted in more than 70 publications and a patent. He guest edited the *IEEE Transactions of Systems, Man and Cybernetics* special issue on unmanned systems and vehicles.

Sood received a Bachelor's degree from the Indian Institute of Technology in Delhi and MS and PhD degrees from Carnegie Mellon University, all in electrical engineering. He is a member of the IEEE Systems, Man, and Cybernetics Society's Administrative Committee.

Address questions concerning this issue to M. Abdelguerfi, Department of Computer Science, University of New Orleans, New Orleans, LA 70148; mahdi@noac.cs.uno.edu.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

High 150

Medium 151

Low 152

VLSI Accelerators for Large Database Systems

VLSI accelerators speed up time-consuming database operations while maintaining the cost and flexibility benefits of general-purpose computers. An experimental VLSI relational data filter performs high-speed associative searches, and a text filter searches for strings at up to 1 Gbyte/s.

Kuo Chu Lee

Takako Matoba Hickey

Victor W. Mak

Gary E. Herman

Bellcore

Attempts to improve the performance of query processing generally fall into one of two categories. Application-specific, or dedicated, database machines¹⁻⁶ improve a specific set of applications and so usually do not support other types of database applications or general-purpose computing. These machines are expensive to develop and maintain, so they are only economically feasible where the cost can be justified.

General-purpose database systems⁷⁻⁹ avoid the problem of specialized hardware by developing database system software on general-purpose computers. But they are less efficient than dedicated machines with customized I/O and processing units.^{4,10,11}

A VLSI accelerator approach can improve the performance of database processing on general-purpose computers. We identified the most time-consuming database operations on general-purpose computers and designed application-specific accelerators to speed up these operations. By using a rapid-turnaround design methodology,¹² we can correctly and efficiently design VLSI accelerators.

Associative search, aggregation, and string search are among the lengthy operations we identified. Associative searches and aggregations are performed on formatted relational records consisting of multiple attributes of specific types and lengths. The associative search selects records

based on a user-defined predicate that specifies the relationships between multiple pairs of attributes and search patterns. The aggregation operation sums or counts an arbitrary attribute of all the selected records. The string search finds all occurrences of a pattern in an unformatted data string.

All of these operations require scanning a large portion of data from the storage devices. As a result, the evolution of mass storage technologies such as optical disks¹³⁻¹⁵ and silicon memory storage¹⁶ has strongly influenced the fundamental design assumptions for VLSI accelerators. These advanced device technologies make possible data transfer rates of up to a few Gbytes/s. At such high I/O rates, direct transfer of data from storage to the CPU is likely to cause thrashing. The CPU cannot perform useful work while it handles frequent page faults caused by the increased I/O operations. Therefore, a critical function of the VLSI search accelerators is to exclude as much irrelevant data as possible before delivering data to the CPU. CPU cycles can be saved for other general-purpose tasks, such as index traversal, updates, transaction processing, and statistical analysis.

We designed and had fabricated an experimental VLSI accelerator research prototype for relational data filtering. This relational data filter performs high-speed associative search and aggregation operations for formatted databases. For unformatted databases, we designed an

experimental fast string search VLSI accelerator that provides a few orders-of-magnitude improvement in text search speed. The VLSI accelerators have precise instruction sets and hardware interfaces that facilitate their integration into general-purpose computer systems and dedicated systems.¹⁷

Formatted database accelerator

The experimental relational data filter speeds up formatted database operations by supporting associative search and aggregation operations. Speeding up the associative search is not a simple task since it requires either a full database scan or a full indexing scheme. A full scan of the database in a general-purpose computer architecture is very costly. Full indexing requires large memory space—at least proportional to the number of data items stored—and incurs large overhead for update and database administration. The full indexing scheme works well for selection queries based on indexed attributes. It is ineffective, however, for complex queries that require scanning a large portion of the database, such as range queries, selection queries with *don't care* characters in the criteria, and queries that take statistical measures on some attributes. Thus, improving the performance of associative searches is a fundamental need in formatted database systems.

We approached the problem of speeding up the full database search operations by using a VLSI relational data filter that scans directly at the high-speed storage output. In support of associative search operations, the data filter selects interesting records from the storage output and passes only those to the main memory. This allows the general-purpose processor to concentrate on more complex operations on the smaller set of records. The data filter also provides high-speed database aggregation operations such as COUNT, SUM, and MAXIMUM, based on arbitrary selection criteria.

Architecture. The VLSI data filter can be viewed as a reduced instruction-set computing (RISC) processor,^{18,19} with the instruction set optimized for associative search and aggregation operations. A more complex comparator, such as an ALU, is needed to enable the filter to perform more complex aggregation operations. However, multiple complex comparators are too costly. The architecture as illustrated in Figure 1 achieves parallelism by efficiently sharing the single comparator unit across multiple queries stored in the instruction buffer.

The data filter accepts records continuously from the storage output, synchro-

nizes the stream speed to the internal speed, and holds each record in an internal record buffer. Records are double-buffered: One record is resident in one buffer and is examined, while the next record is arriving in the other buffer. Instructions are also double-buffered: Instructions in one buffer are executed against a series of records, while the next batch of instructions is loaded into the other buffer. Selected portions of the resident record are evaluated in the execution unit (an ALU and its associated logic) according to the instructions in the instruction buffer. If any of the instruction sequences are evaluated to be true, that is, if the record satisfies a selection query, the ID of the query passes off the chip to the main memory. If any queries are satisfied, the record passes to the main memory at the end of the execution time interval.

The on-chip instruction and record buffers improve the speed of instruction and operand fetch and thus improve the performance of the associative search. Since an associative search requires repetitive application of the same query against the search attributes of all the records in a relation, a reduction in instruction fetch time significantly reduces the overall search time of a single query. Reducing the operand fetch time allows more instructions to be applied against each record; thus, the apparent search parallelism of the data filter is also improved.

Pipelining further improves the apparent execution rate and search throughput by overlapping operations in different stages of the data filter. For example, our prototype

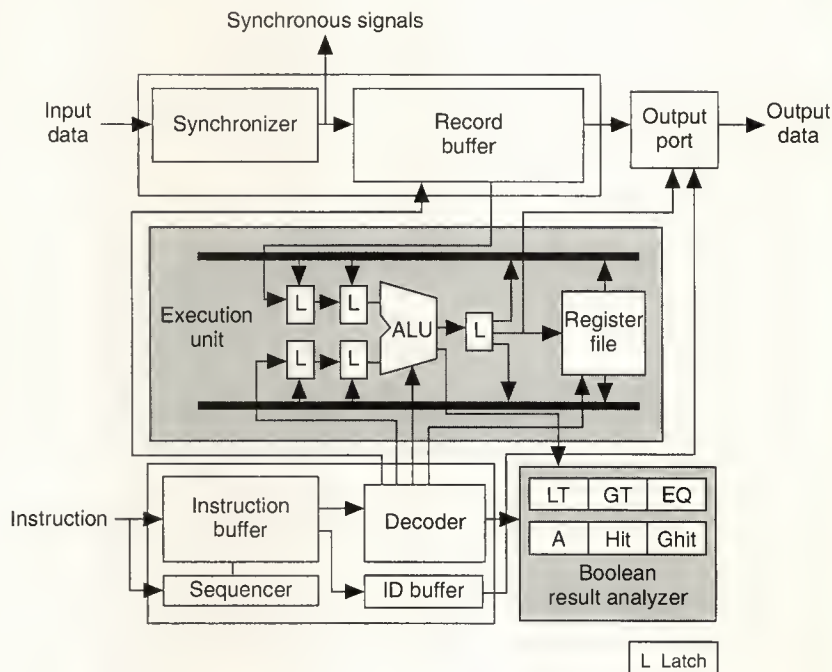


Figure 1. Experimental data filter architecture.

executes each instruction in six pipeline stages including instruction fetch, instruction decode, operand fetch, ALU execution, register store in parallel with Boolean evaluation, and output of query ID.

The size of the record buffer, instruction buffer, and data path can be engineered to suit the application need and technology used. We targeted our prototype for 64 Mbytes/s for the record input stream and 62.5 ns for the processor cycle time. With a 32-bit data path and a 128-byte record buffer, we determined that the instruction buffer holds 32 instructions, the maximum number of instructions that can be executed in the record interarrival time of 2 μ s. A longer record size can be supported by using external buffers. The portion of data specified in the search predicates are stored in the record buffer. We expect that the search attributes for a typical relational record will not exceed 128 bytes.

There are four major blocks inside the data filter.

- **Record buffer control.** The record buffer control synchronizes the arrival of incoming records, manages the double buffering of records, and synchronizes signals for the external control processor. The synchronizer detects only a few special words such as *start of record*, *end of record*, *start of segment*, and *end of segment*. When it finds any of these words, the synchronizer outputs an appropriate boundary indicator and adjusts the address register to place the incoming information in the appropriate location within the record buffer. With minimal embedding of schema information in the data stream, the data filter hardware is free from the runtime decoding of schema information that can slow down filter processing.
- **Instruction buffer control.** The instruction buffer control manages the double buffering of instruction batches, sequencing, and decoding. We kept the logic of instruction buffer control simple by writing a simple instruction set, as discussed later. Without branch instructions, the sequencer becomes a simple counter with a reset logic. The small and orthogonal instruction set enables simple and fast decoder implementation using programmable logic arrays (PLAs).
- **Execution unit.** A parallel data path with an ALU and a register file make up the execution unit. Two operands are fetched into two operand receiving latches and then transferred to the ALU operand latches. By using these two close-coupled latches, the operands receive sufficient set-up time for the latches at the ALU. As a result, the maximum stage delay of the pipe is very close to the clock cycle time if a two-phase, symmetric clock is used. The result of the ALU operation moves from the destination latch to the register file. The operand can be selected from the register file, from the record buffer, or from the pattern specified by the instruction. All the com-

SELECT record FROM relation WHERE boolean expression
FOR ID

where boolean expression ::= term | term OPb boolean
expression

OPb	::= \wedge \vee
term	::= attribute OPa value
OPa	::= $>$ $<$ $=$ \geq \leq \neq

Figure 2. Boolean select operation.

parison and data movement operations are carried through the ALU. In the case of a data dependency, the temporary register can be used to store the result of the first ALU operation and to direct it to the ALU input latch for the next instruction to save translation time and register locking overhead.

- **Boolean result analyzer.** The most important feature of the experimental data filter is the fast Boolean predicate evaluation and the conditional assignment capability defined in the instruction set and supported by the internal architecture. Two single bits, A and Hit, implement the continuous evaluation of conjunctive normal form and conditional assignment that we use to avoid conditional branch instructions, as we discuss later. A comparison result derived from the ALU flags and the test condition is first evaluated with a Boolean accumulator A under the opcode control. Then, the result of the evaluation is stored back into the A accumulator. If the result of the last Boolean operation pertaining to a query is true, a Hit flag is set. This Hit flag is used by the conditional assignment instructions. If it is set, the result of the assignment will be written into the register file; otherwise, the result is discarded.

Instruction set. The instruction set of the data filter blends complex instruction-set computing (CISC) and RISC philosophies. Each instruction exhibits high semantic content and performs multiple actions, as advocated in CISC. However, the number of different instructions is minimized to simplify the instruction decoding, as advocated in RISC. The instruction set has two major modes, one to improve the associative search operations and the other to enable aggregation operations.

Boolean mode. The primary objective of the data filter is to speed up the record selection operations. The data filter takes fixed-size relational records²⁰ as input data and relational queries as instructions to be executed against the input records. Records that satisfy the query predicates become outputs. The basic format of the select operation is given in Figure 2. As an example, the query, "What Indian restaurants are in New York?", may be written in relational algebra.


```

SELECT record FROM (relation = "restaurants")
WHERE (type = "Indian")  $\wedge$  (location = "New York")
FOR (Query 1)

```

The design of the instruction set to realize this selection predicate directly affects search performance. For example, if we assume each string can be coded into one comparison unit, say a long integer, we might implement this query in a general-purpose processor as

```

addr0:  cmp    @relation, #restaurants
        jneq   addr1
        cmp    @type, #Indian
        jneq   addr1
        cmp    @location, #New York
        jneq   addr1
        mov    hit, #1
addr1:  ...

```

Three comparisons correspond to each of the comparisons in the select clause with each followed by a branch instruction.

To improve execution efficiency, the data filter architecture includes Boolean instructions. These instructions speed up the selection operation through the use of the Boolean accumulator A, which accumulates the Boolean result pertaining to one query. Three basic operations can be performed on the Boolean accumulator: PUSHA, which pushes a Boolean value to A; ANDA, which Ands another Boolean value to the value in A; and ORA, which Ors another value to the value in A.

The prototype only supports a one-bit accumulator, but it can be easily extended into a stack to support a wider range of truth operations, such as $(B \wedge C) \vee (D \wedge E)$, where B , C , D , and E are Boolean truth values.

In addition to the three basic operations, four operations on A are defined to tie together the truth values pertaining to one query. FANDA and FORA are like ANDA and ORA but are used for final instructions pertaining to one query. These two instructions turn on the Hit flag, which can be used as a condition for arithmetic instructions discussed in the next section. SANDA and SORA are also used for final instructions, but these turn on the Ghit flag, which causes the output of the current record.

The basic format of a Boolean mode instruction is given by

OPr(OPa(MEM, Pattern, Mask), ID).

where OPr is the operation on accumulator A just discussed. This one instruction specifies three operations to be executed in sequence: a comparison, a test on comparison result, and an operation between the result and the accumulator A. The data filter first compares the word at address MEM (mask)

with Pattern, and, if the comparison flags (LT, GT, EQ) satisfy the flag pattern specified by OPa, the filter sets the Boolean value to true. Finally, it performs the OPr operation on this Boolean value.

A select query may be decomposed into multiple data filter instructions, one instruction for each of the attribute comparisons. For example, the selection query previously presented may be decomposed as

```

PUSHA(EQ(relation, "restaurants", f), 1).
ANDA(EQ(type, "Indian", f), 1).
SANDA(EQ(location, "New York", f), 1).

```

Thus the data filter executes the selection query in only three machine cycles. We intentionally avoid using the exact format here to better illustrate our idea. A similar format can be used as an input to a front-end compiler that compiles the input into the binary form understood by the actual filter. In the example, we use strings such as "relation" to represent memory addresses and use mask f to mean we want the entire attribute.

Arithmetic mode. The second objective of the experimental data filter prototype is to support aggregation operations such as COUNT, SUM, and MAXIMUM. Adding this capability to the filter offloads the main CPU from loading and scanning large amounts of data. It also allows simultaneous execution of aggregation operations that cannot be easily achieved in conventional database systems.

Aggregate operations can be supported in the data filter by adding a conditional assignment capability that allows accumulating the result to a register depending on certain conditions. A conditional assignment is conventionally implemented by a comparison followed by a branch followed by an assignment instruction. However, this approach can be inefficient due to the pipeline breaks. Hence the data filter defines an instruction mode that can perform the ASSIGNMENT operation based on a condition code. This type of instruction together with the Boolean instructions previously discussed allow numeric operations to be carried out without pipeline breaks.

The basic format of the arithmetic mode instruction is given by

OPalu(Cond, M1, M2, M3).

The semantics of the instruction are: If the condition specified by Cond is true, carry out the ALU operation OPALU on two operands M1 and M2 and store the result into M3. The condition can be true (unconditional), hit (set by Boolean instructions), and six combinations of three ALU flags: less-than, greater-than, and equal. M1 can be a memory address or a register, M2 can be an immediate operand or a register, and M3 is a register.

The main loop of the counting query, "How many people live in San Francisco?", can be compiled as

```
PUSHA(EQ(relation, "person", f), 2).
FANDA(EQ(town, "San Francisco", f), 2).
INCB(Hit, null, sumreg, sumreg).
```

A sum query can be formed by replacing the increment operation with an add operation on the summing attribute, say salary, with the sum register.

```
ADD(Hit, salary, sumreg, sumreg).
```

A maximum query can be formed by replacing the increment operation with a compare operation on a maximizing attribute, say age, with the maximum register followed by a conditional assignment of the maximizing attribute to the maximum register.

```
COMP(Hit, age, maxreg, null).
EQUA(Gt, age, null, maxreg).
```

We can extend operations on accumulator A so that the accumulator can be set or cleared based on application-specific condition flags for linking computation results of added functional components. For example, we can add a string search unit, which we describe later, to the data filter as a functional component. The result of a string search operation, finding a pattern in the input stream, can be kept in a condition code and used to set or clear the accumulator.

Discussion. The programmability of the filter lends it to general applicability in a variety of applications requiring high-speed filtering of structured data. Further, the filter architecture scales well with improved fabrication technology, supporting a faster evaluation rate and more complex query sets for finer fabrication geometries.

The VLSI data filter performs better for synchronous search of structured data than both CISC and RISC general-purpose microprocessors. This advantage derives primarily from the architectural support provided both for pipelined Boolean predicate evaluation and for efficient I/O through double buffering of data. We avoided pipeline breaks by using Boolean evaluation and conditional assignment instructions. The parallel data path for instruction and immediate operand store prevents a pipeline stall due to double-operand fetch contention. These features permit more pipeline stages while avoiding pipeline breaks. The architectural optimizations are combined with the overlapped cache refill operations provided by the multiported record buffer and instruction buffers. Together they allow the VLSI data filter to perform synchronous data search faster than a conventional microprocessor used in the same technology.

We made the data filter more efficient by integrating sev-

eral subsystems on a single VLSI chip. The experimental VLSI data filter includes additional subsystems beyond the microprocessor itself, including a tri-port RAM for the record buffer, dual-port RAM for the instruction buffer, and extensive glue logic to support synchronization, instruction sequencing, and handshaking. Implemented off chip, these functions would require expensive high-speed components to perform as well as the integrated design. We estimate that it would take four times as much hardware to replicate the data filter and its support functionality compared to our prototype design. The improvements in execution efficiency, combined with the factor-of-four reduction in hardware, provide a bias for advocating the use of a VLSI accelerator over an approach based on a general-purpose microprocessor.

Designers can achieve the performance advantage with only a modest effort by taking advantage of high-level specification languages and VLSI tools. Matoba et al.¹² describe a rapid-turnaround methodology that we used to realize our data filter prototype in nine staff-months. The ability to design quickly a VLSI system as complex as the data filter should allow custom VLSI to be commercially viable in many more application systems than it has been. The ability to explore options rapidly with precision should allow systems research to progress rapidly beyond the paper design stage where ideas often stall for lack of expertise and/or human resources.

The VLSI research prototype fabricated in 2- μ m CMOS technology executes more than 16 million predicate evaluation instructions per second and achieves 64-Mbyte/s search throughput for individual queries or sets of queries comprising a total of up to 30 Boolean predicates. Table 1 summarizes some chip parameters for our first prototype. The chip consists of 91,000 transistors and the area is 475 mils square. The chip's speed is determined by the longest stage in the pipeline (54 ns, determined by the speed of the ALU cell). The first wafer lot produced 33 working chips, a yield of 16 percent. The prototype filter is an integral part of a working experimental prototype database system that efficiently supports a rich set of query types plus full transaction semantics.¹⁷

Table 1. Experimental chip parameters.

Number of transistors	91,000
Chip size	476 \times 477 mil ²
Number of pins	171
Cycle time	54 ns
Power at 18.5 MHz	2.5 W
Supply voltage	5 V
Process technology	2 μ m CMOS

Unformatted string search accelerator

String searching is another operation that requires extensive scanning of unformatted data in database applications. The problem of string searching is to find all occurrences of a p -character pattern P constructed from a vocabulary of m distinct characters in an s -character data string S . The pattern P may also contain *don't care* characters. For typical applications, $p \ll s$ and $m \ll s$. Since the size of the data string is usually very large, sequential search via general-purpose processors is prohibitively slow. Thus, reducing the time required to search a large text database has been an active area of research for the past decade.

We propose a new parallel VLSI algorithm called Data Parallel Pattern Matching (DPPM) and a corresponding VLSI document-search engine called DPPME. The DPPM algorithm differs from most previous work in that it serially broadcasts each character in the pattern and compares the pattern to a block of data in parallel. The DPPM algorithm uses the high degree of integration of VLSI technology to process quickly through parallelism. Based on simulation statistics and timing analyses of the hardware design, we can achieve a search rate of multiple Gbytes/s DPPME using advanced VLSI technology.

The DPPM algorithm. Let $S[1:n]$ be the data string of n characters to be searched and $Pat[1:p]$ be the pattern of p characters. The data string is divided into blocks of b characters each and searched a block at a time. Let $Blk[1:b]$ be the current data block of size b characters. Basically, the DPPM

algorithm serially broadcasts each pattern character to a block of the data string. If the pattern character matches with any of the characters in the block, the next pattern character is broadcast in the next comparison cycle. If, at any cycle, no match is found between the current pattern character and the data block, and if no partial match is carried over from the previous block, the filter discards the data block. The search continues with the next block. A partial match occurs when $Pat[i]$, $i < p$, matches with the last data block character $Blk[b]$. This partial match information is stored and used in the next block to continue the search by comparing $Pat[i+1]$ to the first data block character $Blk[1]$.

We can best illustrate the DPPM algorithm with a simple example. Suppose we conduct a search for the pattern "abcd" in the data string "abadbbabcedee." Figure 3 shows the operation of the DPPM algorithm using a block size of four.

The first block, containing the characters "abad," is first loaded into the comparator array. When compared to the first pattern character "a," two matches are detected. The second pattern character "b" is then broadcast and compared to the characters to the right of the matched characters in the previous cycle, that is, "b" is compared to the second and the fourth characters in the block. Since a match is detected again at the second character, a comparison of the third pattern character "c" with the third character in the block is necessary. This time, no match in the block is observed; thus, the current block is discarded and the search continues with the next block. This early mismatch detection mechanism avoids broadcasting and comparing the fourth pattern character "d" to the current block since this comparison is unnecessary.

The next block contains the characters "bbab." The pattern compares successfully up to the second character "b." At this point, the end of the block is reached. The pattern may span the block boundary. DPPM acknowledges this partial match information and continues the match in the next cycle. Since no other match occurs in this block, the current block is also discarded.

The third block contains the characters "cedee." The first pattern character "a" has no match with the block. At this point, DPPM acknowledges a partial match in the previous block up to the second pattern character. Therefore, it jumps to the third pattern character "c" and continues the partial match from the previous block. Finally, a hit or an occurrence of the pattern is detected with the fourth pattern character "d."

Although not shown in this example, the DPPM algorithm can also detect multiple occurrences of the pattern even if they overlap, and no backtracking is required to detect all occurrences.

Figure 4 on page 14 shows the pseudocode of the control flow of the DPPM algorithm. $DC[1:p]$ is a bit-vector indicating the *don't care* positions in the pattern. $DC[i]$ is set if there is a *don't care* at $Pat[i]$. $Mask[1:b]$ controls the activation of the

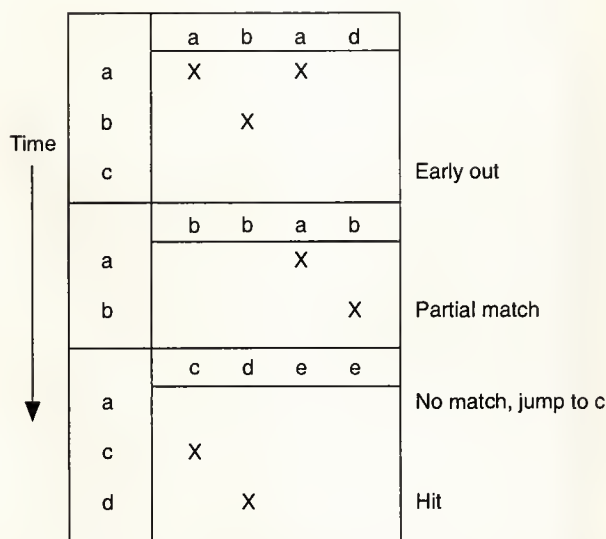


Figure 3. DPPM example.

comparator array based on the results of the previous comparison cycle. If $Pat[i]$ matches $Blk[i]$ in the first comparison cycle, then $Mask[i+1]$ is set in the next comparison cycle enabling the comparison between $Pat[2]$ and $Blk[i+1]$. $T[1:b]$ holds the results of the comparator array. $Vin[2:p]$ and $Vout[1:p-1]$ hold the partial match information from the previous block and the current block, respectively. $Vin[i]$ is set if a partial match is found in the previous block up to and including the character $Pat[i-1]$. $Vout[i]$ is set if a partial match is found up to and including the character $Pat[i]$ in the current block.

With the use of Vin and $Vout$, partial match can be continued in the next block without any backtracking of the data string. Since the algorithm is independent of the block size, the search rate can be increased simply by increasing the block size, or the number of comparators. As a result, the DPPM algorithm can efficiently use the high degree of integration of

```

while (← End of Data String) do begin
  GetNextBlock(Blk);
  forall i from 1 to b do Mask[i] := TRUE;
  forall i from 1 to (p-1) do Vout[i] := FALSE;
  i := 1;
  while i ≤ p do begin /* Comparison Cycle */
    forall j from 1 to b do
      T[j] := Mask[j] ∧ (DC[i] ∨ (Blk[j] = Pat[i]));
    if (i < p) then Vout[i] := T[b]
    else begin /* i = p: Report hit found */
      forall j from 1 to b do
        if (T[j]) then Report Hit at j;
      break; /* Match(es) in block */
    end;
    if (Vj=1p-1 T[j]) then begin
      /* Prepare Mask for next comparison cycle */
      forall j from 2 to b do
        Mask[j] := T[j-1];
      Mask[1] := Vin[i+1];
      i := i + 1;
    end
    else if (Vj=i+1p Vin[j]) then begin /* Continue partial match */
      /* Skip Unnecessary Pattern Characters */
      do i := i + 1 until (Vin[i]);
      Mask[1] := TRUE;
      forall j from 2 to b do Mask[j] := FALSE;
    end
    else /* Early Out */
      break;
  end;
  forall i from 2 to p do Vin[i] := Vout[i-1];
end;

```

Figure 4. Pseudocode of the DPPM algorithm.

VLSI technology to attain high-speed processing through parallelism.

VLSI prototype design. Figure 5 shows the circuit block diagram of the experimental DPPM engine. Before the actual search operation, the pattern, pattern length, and *don't care* (DC) positions are first loaded into their corresponding registers. The data string is buffered and read one block at a time to the Block register. The comparator array performs the actual comparison between the pattern character and the data block. The results of the comparator array are Anded with the mask to form register T. The DPPM engine integrates the mismatch detection and the partial match propagation mechanisms by combining the Vin register (partial match information from previous block) with the old T register (match results of the previous comparison cycle) to form the mask for each comparison cycle. The first bit of the mask is from $Vin[i]$, and the last (b-1) bits are from the first (b-1) bits of T in the previous cycle. The last bit of T is stored into $Vout[i]$. Each time a new data block is read, the first (p-1) bits of $Vout$ are loaded into the last (p-1) bits of Vin . The first bit of Vin is always set.

The sequence controller controls the operation of the DPPM engine by generating the value i , which is used to index the pattern, *don't care*, Vin , and $Vout$ registers. By monitoring the values of T, the content of the Vin register, and the pattern length, the sequence controller decides for each cycle one of the following three actions:

- Compare the next pattern character with the current block,
- Jump to a pattern character to continue the partial match from the previous block, or
- Discard the current block and continue the search with the next block.

Step 1 is taken if this is not the last pattern character and the content of T, that is, the number of matches detected in the current cycle, is nonzero. If T is zero, then the index of the next pattern character to be used for comparison is determined by a priority encoder that encodes the first nonzero bit in the Vin register after masking off the first (i-1) bits of the Vin register using a linear shift register. Step 3 is taken if the last pattern character is reached, or if T is zero and there is no more partial match propagation from the previous block (early out).

The sequence controller also generates a last character signal when the last pattern character is reached. This signal is used by the hit detection unit, which checks the values of T to report any hit in the search. The priority encoder produces the encoded addresses for all hit positions in the current block.

The critical path of the circuit is from the pattern register, through the comparator and And arrays, to the priority encoder. Using 2- μ m CMOS technology, we achieve the comparison cycle time of about 50 ns for a block size of 1,000. This cycle time includes the broadcasting delay of 6 ns. If 1.2- μ m

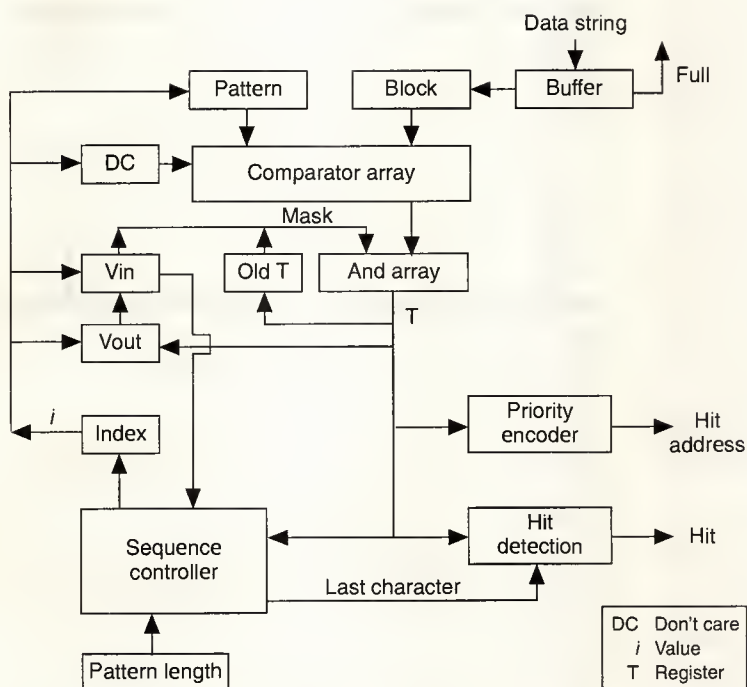


Figure 5. Circuit block diagram of the experimental DPPM engine.

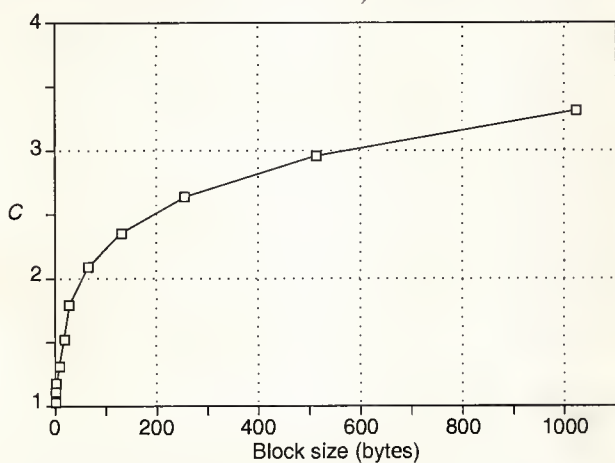


Figure 6. Average number of comparison cycles per block.

CMOS is used, the cycle time will be approximately 33 ns. The chip area of the DPPM engine for a block size of 128 is roughly $200 \times 100 \text{ mil}^2$.

It is important to understand whether the I/O bandwidth of a CMOS chip can sustain the Gbyte/s search rate of the DPPM engine. Marcus²¹ has demonstrated that using 32 input pads, a bandwidth of 5.44 Gbytes/s can be achieved with 1.2- μm

CMOS technology. We can achieve even higher bandwidth by using advanced packaging technology that supports higher I/O pin counts.²²

Performance evaluation. We can evaluate the DPPM algorithm by simulating its operation on a real text database. Under such an environment, we can estimate its performance, and also learn about its behavior. We chose as a database the Associated Press wire news articles on August 2, 1988, a 4.4-Mbyte database. For test patterns, we chose the 100 most frequently occurring words in American English that are at least five characters long. The pattern lengths vary from 5 to 10 characters with an average of 5.88 characters.²³

Figure 6 shows the average number of comparison cycles per block C , measured at different block sizes. Although the pattern characters are serially compared to the data block, early mismatch detection allows the algorithm to search the next block as soon as a mismatch is detected. This feature is especially effective at smaller block sizes where the probabilities of matching the first few characters of the pattern are low. Without early mismatch detection, C equals the average pattern length, in this case 5.88. At larger block sizes, the probability of finding the pattern in the block is higher, thus the value of C also increases.

As the block size increases, the value of C approaches the average pattern length asymptotically.

Using 50 ns as the comparison cycle time T , Figure 7 shows the search rates R_b at different block sizes. Recall that increasing the block size requires proportionately more comparators on a chip. At a block size of 16, the search rate is 212 Mbytes/s. This rate matches the predicted optical disk trans-

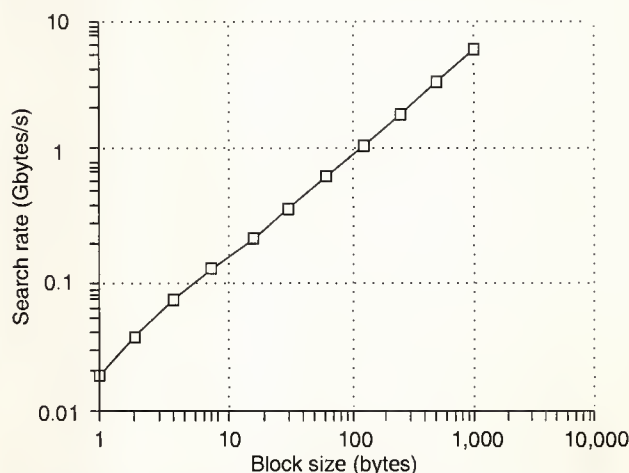


Figure 7. Search rates at different block sizes.

fer rate of 200 Mbytes/s.¹³ At a block size of 128, the search rate reaches 1 Gbyte/s. This rate is sufficient to handle existing memory bandwidth of supercomputers as well as data input from optical-fiber transmission systems in future communication networks.

Extension and integration. Instead of comparing the pattern characters serially to the data block as in the DPPM algorithm, it is possible to compare multiple pattern characters to the data block. The Multiple-Pattern, Multiple-Data (MPMD) approach reduces the number of comparison cycles required per block and thus results in a higher search rate. Since multiple-pattern characters can be grouped into a word with proper length (for example, four characters long for a 32-bit processor), the MPMD engine can be easily integrated with the relational data filter previously described so that the search operation can be performed on both formatted and unformatted data.

An MPMD engine uses a 2D array of comparators to perform multiple steps of the DPPM algorithm in parallel. Pattern characters can be loaded and compared in parallel to the input data block. The partial match traces propagate through the comparators in a diagonal direction and terminate at the V registers for partial matches and T registers for matches. The whole parallel comparison operation can be implemented in one or more pipeline stages. Notice that the maximum length of the critical path equals the minimum length of the input block and the pattern.

To integrate the MPMD engine in the experimental relational data filter, the designer must provide interfaces to pass the control information between the MPMD and the data filter instruction set. A Match flag can be used to indicate one or more matches terminated at the current input data block (indicated in the T register). This Match flag can be used as a condition code for Boolean predicate evaluation instructions. Furthermore, since all the state information of the string search operation is stored in the V and T registers, these registers can be mapped as a part of the register file. Since the string search operation is performed while the data is loaded into the relational data filter, the search results for each input record can be stored in flags and registers. The text is also segmented into 128-byte records. The results are then forwarded to the relational filtering operation in pipelined fashion. The relational data filter can then use and manipulate directly the search results stored in the condition codes and in the general-purpose registers.

The speed of the MPMD circuit is fast enough to implement in one pipeline stage that matches the VLSI chip I/O rate. The critical path of the MPMD circuit for an input data block size of 4 bytes wide (8 bits per byte) consists of a byte comparator plus four stages of combinational logic for partial match propagation. Using a conservative VLSI technology with about 300-ps gate delay time, the byte comparison will take about 5 ns (including pattern and data broadcast time,

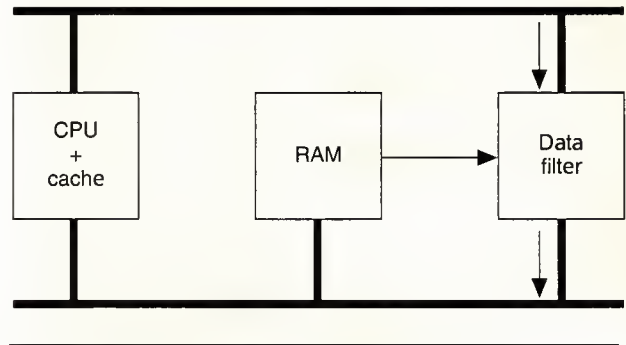


Figure 8. Accelerator and processor integration.

XOR comparison logic, and eight carry propagation stages.) The partial match signal propagating through four stages (4-byte comparator in diagonal) will take about 1.2 ns. Thus, the total time of each parallel comparison will be less than 8 ns. Since this comparison speed matches the maximum data input rate of high-performance I/O pads,²¹ it is unnecessary to introduce extra pipeline stages for the comparator array.

For concurrent query evaluation, we can construct multiple-parallel MPMD building blocks that allow multiple text patterns to be searched simultaneously. To do that, multiple Match flags can be indexed for further Boolean predicate evaluation. With current VLSI technology, we estimate that thirty-two 8-byte-long text patterns can be compared in parallel (total 1,000-byte comparators) in one VLSI chip. This estimation is conservative; however, a small VLSI chip may significantly increase the yield and reduce the cost. Such an accelerator can search 32 text patterns concurrently at about 500-Mbyte/s input data rates.

VLSI accelerators can achieve very fast string searches, at rates easily exceeding the I/O data rate of VLSI chips. The density of current VLSI technology can further improve the throughput of concurrent string searches. The test string matching operations can operate synchronously with the Boolean predicate evaluation operations used in data scan operations.

Application notes

The proposed experimental VLSI accelerators can enhance the performance of general-purpose computing and dedicated systems.

General-purpose computer systems. We use an abstract data processing model to illustrate the applications of the experimental VLSI accelerators for general-purpose computing systems. The abstract model consists of a set of storage devices and a set of VLSI search accelerators interconnected by a high-performance bus interconnection network. This network provides fast virtual circuit connections for communication sessions required by the applications. As shown in Figure 8, designers can easily incorporate the accelerators

into a board-level processing node. A typical processing node consists of a CPU with high-speed caches as well as a large set of dual-ported main memory. The accelerators process data on a page-by-page basis. The data comes through the bus from either the main memory or other nodes. While the data filter is executing the search and aggregation operations, the CPU can execute other tasks. When the search and aggregation operations are completed, the search results can be accessed by the CPU through the main memory.

For a parallel computer, multiple processing nodes are connected by an interconnection network (the design of which is beyond the scope of this article). A search operation starts from the loading of search instructions into one or more data filters. Communication channels between the data sets and the accelerators are established to form a filter network. After the filter network is formed, the data passes through the network in a pipelined fashion. The input and output of each data filter are buffered synchronously or asynchronously, depending on the applications. Figure 9 shows a tree structured associative search task designed to select data records satisfying the predicate ((C and A) or (C and B)), where A, B, C are patterns. The first two accelerators search for two disjunctive patterns A and B, respectively, from two data sets. The third accelerator searches for pattern C from records selected by the first two filters.

Dedicated systems. The experimental VLSI data filter is a critical component in a large-scale broadcasting database system prototype called the Databycle architecture.¹⁷ The Databycle architecture demonstrates the feasibility of providing a high-performance and highly flexible database access services for a large number of users in geographically distributed areas. Databycle consists of a set of data and data filters located in multiple distributed sites. The system broadcasts data to data filters cyclically through optical fibers. (To support applications with a very large volume of data, only the

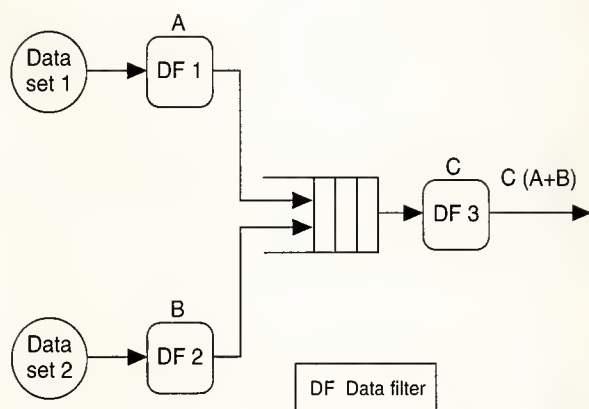


Figure 9. A filter network.

part of the data that is constantly changing or frequently accessed by multiple sites are broadcast repetitively.) Search queries are translated into data filter instructions and distributed to multiple data filters that execute them concurrently.

Since the data is made available to all the data filters simultaneously, the system can concurrently process an unlimited number of queries. This broadcast property is fundamental in supporting concurrent associative search, aggregation, and transaction processing services for millions of users on a set of constantly changing data.

To demonstrate its implementation feasibility, we installed a prototype system of the Databycle architecture in our laboratory. It consists of silicon memory-based data storage devices, 500-Mbps fiber-optic broadcasting links, and multiple formatted data filter boards. The architecture of a distributed site containing data sets is similar to the general-purpose node architecture shown in Figure 8. A distributed site that performs only data accessing contains three VLSI data filters and associated controllers. The prototype system demonstrated the functionality of the Databycle architecture. An extended query set can be programmed with the data filter instructions. These extended queries execute in two phases. First, the data filter prefilters to reduce the amount of data to be processed later. Second, the general-purpose CPU completes the query processing and retrieves the data.

We learned several lessons from the Databycle prototype. First, the effort involved in the VLSI accelerator design is very small compared to the total effort involved in software development and system integration (less than 20 percent). This is partly due to the advancement in CAD tools but mostly due to the increasing complexity of system and application software. Second, the use of VLSI accelerators significantly improves the performance of the overall system and enables new capabilities more economically.

Frieder, Lee, and Mak proposed a document-searching architecture, based on the DPPM engines, to increase the throughput of an information retrieval system.^{23,24} The proposed architecture (Figure 10 on page 18) is comprised of a set of DPPM engines and a single master processing element (PE). An information retrieval query is decomposed by the PE into basic match primitives to be executed in the DPPM engines, while the query (a sequence of operators) is evaluated at the PE using results from the document-search engines. By separating the operator and query complexity from the DPPM engines, the complexity of the customized hardware is freed from the complexity of the query.

The PE instruction set is based on the text retrieval machine instruction set presented by Hollaar et al.²⁵ but modified to be match-based. In other words, each instruction is defined as a set of match conditions with a simple imposed control structure. The instruction set²³ supports Boolean operations of patterns as well as wild-card characters that match one or more characters in the pattern. As the imposed

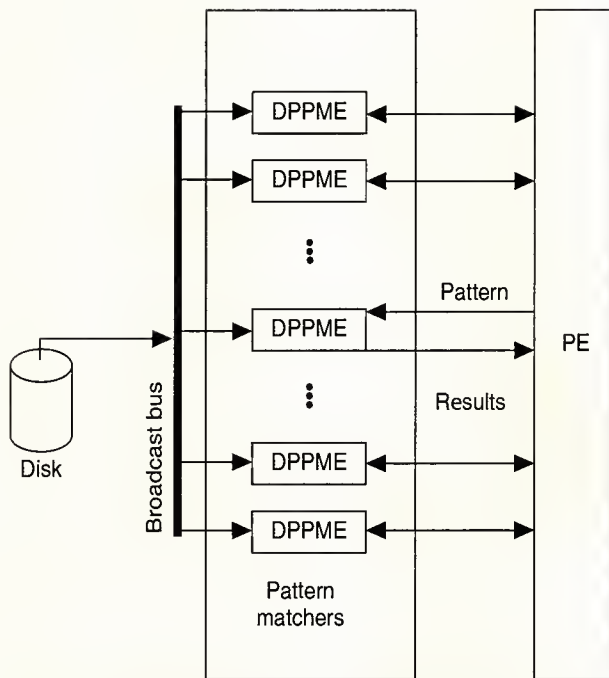


Figure 10. Proposed document-searching architecture.

control structure requires minimal computation time relative to the amount of processing involved in searching the document stream, it can be implemented in software without significantly affecting the system throughput. A 50-MIPS PE can support 10 DPPM engines of 128 blocks in parallel, at a utilization level of 0.25, with each DPPM engine having a search rate of 1 Gbyte/s. At this search rate, we can search both the Old and the New Testaments of the Bible in 5 ms, *Webster's Dictionary* in 16 ms, and the entire volume of *Encyclopaedia Britannica* in 400 ms.

Related work

Numerous hardware-based solutions have been proposed and implemented to expedite both the associative and string search operations. As fast software string-searching algorithms^{26,27} are based on finite-state automata (FSA), hardware realizations of FSA pattern matching were investigated by several researchers.²⁸⁻³⁰ FSA requires precompilation of the patterns and processes the data string one character at a time. Although precompilation of the pattern eliminates the need to compare each character of the data string to every pattern character, the sequential character-at-a-time processing severely limits the search rates of these systems.

Several researchers³¹⁻³⁴ propose using comparator arrays to perform pipelined pattern matching directly without precompilation of the patterns. Multiple patterns are com-

pared concurrently to the data string to achieve higher throughput. However, the search rate is still limited by the sequential processing of the data string. In the systolic array approach,³¹ data and pattern characters are routed in opposite directions. At any given clock cycle, only half of the cells in the array can perform meaningful computation. Therefore, half of the physical hardware is actually wasted. In some proposals,^{32,34} pattern characters are preloaded into the comparator cells. Each character of the data string is broadcast into all cells serially, and comparison results are generated by all cells simultaneously. Since a string search operation on text database exhibits very low selectivity, comparisons beyond the first few characters of the pattern are usually unnecessary. Thus, most of the comparisons with the last few characters of the pattern are redundant.

To reduce the number of redundant comparisons and to increase the degree of effective parallelism in the pattern matching problem, both Altep³⁵ and our string-search algorithm use a data parallel, pattern serial scheme in which pattern characters are broadcast and compared to a block of the data string in parallel. While Altep is a cellular processor optimized for regular expression comparisons with microprogrammed control, our DPPM engine is a VLSI filter optimized for variable-length text processing with hard-wired control. The decoupling of query resolution from the primitive match operation simplifies the structure of the DPPM engine so that it can be implemented compactly and is hence more efficient.

Relational data filtering is different from unformatted string searching in that in the former, the starting position of the pattern to be matched is specified in the record format definition. As a result, associative search on relational data requires fewer comparison operations than string search. Another difference is that relational data filtering requires more complex search predicates than does unformatted string searching. Therefore, the filters designed for string search are not efficient for relational data filtering.

The proposed relational data filter differs from other filters^{28,29,36-39} in that it is a RISC architecture with a highly programmable instruction set and efficient implementation. As indicated earlier, the instruction set of the data filter can construct a wide mixture of associative search and aggregation queries based on efficient Boolean predicate and conditional assignment operations. The simplicity of the reduced instructions also results in fast VLSI implementation.

THE PROPOSED DATABASE ACCELERATOR can offload the CPU from the time- and space-consuming associative search and aggregation operations. The novel instruction set and Boolean accumulator support allow highly pipelined Boolean predicate evaluation for on-the-fly processing over high-speed input data streams available in the future storage technology. The efficient VLSI architecture results in a 64-

Mbyte/s associative search rate via a simple design with modest fabrication technology. The architecture can be easily extended to support a higher data rate by increasing the width of the data path and the number of pipeline stages. The VLSI data filter achieves significant performance advantages for synchronous searches of formatted data when compared with general-purpose microprocessors. Its efficiency increases when we integrate several subsystems onto a single VLSI chip.

The VLSI DPPM engine overcomes the fundamental limitation of sequential pattern-matching algorithms and efficiently processes consecutive text strings in variable block size. We can achieve a search rate of Gbytes/s with simple VLSI implementation of the algorithm. By incorporating the DPPM engine with high-speed data channels available in silicon memory and optical storage devices, we can economically resolve the problem of slow response time for large information service databases.

Through the actual design and the use of VLSI accelerators, we learned that VLSI accelerators are a very cost-effective means for improving overall system performance. Modern VLSI design tools can significantly reduce the time and the cost believed to be required for custom IC design. As a consequence, the VLSI accelerator approach is a relatively low-risk and inexpensive one when the overall software and hardware system design and integration costs are considered. Additionally, we learned that the value of VLSI accelerator design is not limited to the system performance gain. Basic functional components and I/O structures abstracted from the accelerator designs can potentially become standard features for future general-purpose CPU architecture designs. ■

Acknowledgments

We acknowledge Ophir Frieder for his contribution to various aspects of the DPPM work. We also acknowledge Tony McAuley, Yow-Jian Lin, Jane Cameron, and Bill Mansfield for their valuable comments, which significantly improved the quality of this article.

References

1. C.K. Baru and S.Y.W. Su, "The Architecture of SM3: A Dynamically Partitionable Multicomputer System," *IEEE Trans. Computers*, Vol. C-35, No. 9, Sept. 1986, pp. 790-802.
2. D.K. Hsiao, ed., *Advanced Database Machine Architectures*, Prentice Hall, Englewood Cliffs, N.J., 1983, pp. 1-18.
3. W. Kim, D. Gajski, and D. Kuck, "A Parallel Pipelined Relational Query Processor," *ACM Trans. Database Systems*, Vol. 9, No. 2, June 1984, pp. 214-242.
4. M. Kitsuregawa, M. Nakano, and M. Takagi, "Query Execution for Large Relations on Functional Disk System," *Proc. Fifth Int'l Conf. Data Engineering*, IEEE Computer Society Press, Los Alamitos, Calif., 1988, pp. 159-167.
5. E. Ozkarahan, *Database Machine and Database Management*, Prentice Hall, Vol. 10, 1986, pp. 319-337.
6. *DBC/1012 Data Base Computer: Concepts and Facilities*, C02-0001-05 Release 3.1, Teradata, Inc., 1989.
7. C.K. Baru and O. Frieder, "Database Operations in a Cube-Connected Multicomputer System," *IEEE Trans. Computers*, Vol. 38, No. 6, June 1989, pp. 920-927.
8. D. DeWitt et al., "GAMMA—A High Performance Dataflow Database Machine," *Proc. VLDB Conf.*, Morgan Kaufmann Publishers, Los Altos, Calif., 1986, pp. 228-237.
9. M. Smith et al., "An Experiment on Response Time Scalability in Bubba," *Proc. Int'l Workshop on Database Machines*, Springer-Verlag, New York, 1989, pp. 34-57.
10. M. Kitsuregawa et al., "Design and Implementation of High Speed Pipeline Merge Sorter With Run Length Tuning Mechanism," in *Database Machines and Knowledge Base Machines*, M. Kitsuregawa and H. Tanaka, eds., Kluwer Academic Publishers, Boston, 1988, pp. 89-102.
11. C. Lee, S.Y. Su, and H. Lam, "Algorithms for Sorting and Sort-Based Database Operations Using a Special-Function Unit," in *Database Machines and Knowledge Base Machines*, M. Kitsuregawa and H. Tanaka, eds., Kluwer Academic Publishers, 1988, pp. 103-116.
12. T. Matoba et al., "A Rapid Turnaround Design of a High Speed VLSI Search Processor," *Integration, VLSI J.*, Vol. 10, 1991, pp. 319-337.
13. P.B. Berra and N.B. Troullinos, "Optical Techniques and Data/Knowledge Base Machines," *Computer*, Vol. 20, No. 10, Oct. 1987, pp. 59-70.
14. S. Redfield and L. Hesselink, "Data Storage in Photorefractives Revisited," *Proc. Conf. on Optical Computing*, Vol. 963, 1988, pp. 35-45.
15. T.A. Shull, R.M. Holloway, and B.A. Conway, "NASA Spaceborne Optical Disk Recorder Development," *Optical Storage Technology and Applications*, Vol. 899, 1988, pp. 272-278.
16. H. Garcia-Molina, R.J. Lipton, and J. Valdes, "A Massive Memory Machine," *IEEE Trans. Computers*, Vol. C-23, No. 5, May 1984, pp. 391-399.
17. G.E. Herman et al., "The Datacycle Architecture for Very High Throughput Database Systems," *Proc. ACM-SIGMOD Conf.*, ACM, N.Y., May 1987, pp. 97-103.
18. J. Hennessy, "VLSI Processor Architecture," *IEEE Trans. Computers*, Vol. C-33, No. 12, Dec. 1984, pp. 1221-1245.
19. S.A. Przybylski et al., "Organization and Implementation of MIPS," in *Tutorial on Advanced Microprocessors and High-Level Language Computer Architecture*, V. Milutinovic, ed., IEEE CS Press, 1984, pp. 202-239.
20. C.J. Date, *An Introduction to Database Systems*, Addison-Wesley, Reading, Mass., 1986.
21. W. Marcus, "A CMOS Batcher and Banyan Chip Set for B-ISDN

- Packet Switching," *IEEE J. Solid-State Circuits*, Vol. 25, No. 6, 1990, pp.1426-1432.
22. R. Johnson, "Multichip Modules: Next-Generation Packages," *IEEE Spectrum*, Vol. 27, No. 3, Mar. 1990, pp. 34-48.
 23. V.W. Mak, K.C. Lee, and O. Frieder, "Exploiting Parallelism in Pattern Matching: An Information Retrieval Application," *ACM Trans. Information Systems*, Vol. 9, No. 1, Jan. 1991, pp. 52-74.
 24. O. Frieder, K.C. Lee, and V.W. Mak, "JAS: A Parallel VLSI Architecture for Text Processing," *Data Engineering*, Vol. 12, No. 1, Mar. 1989, pp. 16-22.
 25. L.A. Hollaar et al., "Architecture and Operation of a Large, Full-Text Information Retrieval System," in *Advanced Database Machine Architecture*, D.K. Hsiao, ed., Prentice Hall, 1983, pp. 256-299.
 26. R.S. Boyer and J. Moore, "A Fast String Searching Algorithm," *Comm. ACM*, Vol. 20, Oct. 1977, pp. 762-772.
 27. D.E. Knuth, J.H. Morris, Jr., and V.R. Pratt, "Fast Pattern Matching in Strings," *SIAM J. Computing*, Vol. 6, June 1977, pp. 323-350.
 28. F. Bancilhon et al., "Verso: A Relational Backend Database Machine," in *Advanced Database Machine Architectures*, D.K. Hsiao, ed., Prentice Hall, 1983, pp. 1-18.
 29. R. Gonzalez-Rubio, J. Rohmer, and D. Terral, "The Schuss Filter: A Processor for Nonnumerical Data Processing," *Proc. ACM SIGARCH Conf.*, 1984, pp 64-73.
 30. R.L. Haskin and L.A. Hollaar, "Operational Characteristics of a Hardware-Based Pattern Matcher," *ACM Trans. Database Systems*, Vol. 8, Mar. 1983, pp. 15-40.
 31. M.J. Foster and H.T. Kung, "The Design of Special Purpose Chips," *Computer*, Vol. 13, No. 1, Jan. 1980, pp. 26-40.
 32. A. Halaas, "A Systolic VLSI Matrix for a Family of Fundamental Search Problems," *Integration, VLSI J.*, Vol. 1, Dec. 1983, pp. 269-282.
 33. H.T. Kung and P.L. Lehman, "Systolic VLSI Array for Relational Database Operations," *Proc. ACM SIGMOD Conf.*, May 1980.
 34. K. Takahashi, H. Yamada, and M. Hirata, "Intelligent String Search Processor to Accelerate Text Information Retrieval," *Proc. Fifth Int'l Workshop on Database Machines*, Kluwer Academic Publishers, Oct. 1987, pp. 440-453.
 35. D. Lee, "Altep—A Cellular Processor for High-Speed Pattern Matching," *New Generation Computing*, Vol. 4, Sept. 1986, pp. 225-244.
 36. P. Faudemay and P. Valduriez, "Design and Analysis of a Direct Filter Using Parallel Comparators," *The Fourth Int'l Workshop on Database Machines*, D.J. DeWitt and H. Boral, eds., Springer-Verlag, Mar. 1985.
 37. G. Gardarin et al., "SABRE: A Relational Database System for a Multimicroprocessor Machine," D.K. Hsiao, ed., *Advanced Database Machine Architectures*, Prentice Hall, 1983, pp. 19-35.
 38. H. Schweppe et al., "RDBM—A Dedicated Multiprocessor System for Database Management," D.K. Hsiao, ed., *Advanced Database Machine Architectures*, Prentice Hall, 1983, pp. 36-85.
 39. S.Y.W. Su et al., "The Architecture Features and Implementation Techniques of the Multicell CASSM," *IEEE Trans. Computers*, Vol. C-28, No. 6, June 1979, pp. 430-445.



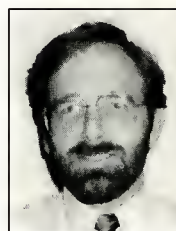
Kuo Chu Lee is a researcher in the Applied Research Area of Bellcore in Morristown, New Jersey. His interests include distributed cooperative systems, extended transaction management systems, and design of large-scale multimedia systems. Lee received a BS in power mechanical engineering from National Tsing-Hua University, an MS in electrical engineering from Ohio State University, and a PhD in computer engineering from Rutgers University. He is a member of the IEEE.



Takako Matoba Hickey is a researcher in the same area. Her research interests include fault-tolerant computing, distributed systems, and software engineering. She received a BS degree in engineering and applied science from California Institute of Technology and an MS in computer science from Stanford University.



Victor W. Mak is also a researcher in the Applied Research Area of Bellcore. His research interests include distributed systems, parallel processing, object-oriented programming, and performance evaluation. Mak received a BS degree in electrical engineering from the University of Toronto and MS and PhD degrees in electrical engineering from Stanford University. He is a member of the IEEE and ACM.



Gary E. Herman is division manager of Network Systems Research in the Applied Research Area of Bellcore. He received the BSE and PhD degrees in electrical engineering from Duke University.

Direct questions concerning this article to K.C. Lee at Bellcore, 445 South Street, Morristown, NJ 07962-1910; or via e-mail at kasey@thumper.bellcore.com.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 153

Medium 154

High 155



An Associative Accelerator for Large Databases

The RAPID-1 (Relational Access Processor for Intelligent Data) is an associative accelerator that recognizes tuples and logical formulas. It evaluates logical formulas instantiated by the current tuple, or record, and operates on whole relations or on hashing buckets. RAPID-1 uses a reduced instruction set and hardwired control and executes all comparisons in a bit-parallel mode. It speeds up the database by a significant factor and will adapt to future generations of microprocessors.

Pascal Faudemay

Mongia Mhiri

Laboratoire MASI

Information servers should perform at a higher level to be usable for more numerous applications. Among such applications are real-time information systems; large knowledge bases; and very large databases, such as those that hold seismic, textual, and financial data. A solution to the performance problem may come from better algorithms, more parallel systems, or more calculating power in the processor. In any case, good processing power is desirable and may reduce the complexity of parallel machines or help to improve performance on sequential machines.

To increase the power of general processors, we can rely on processors themselves, or on application-specific integrated circuit (ASIC) accelerators. The old reasoning that specific circuits, which were not yet ASICs, cannot perform as well as general circuits contradicts all research on ASIC and ASIC-dedicated CAD.¹⁻³ Today's ASIC circuits are designed at the schema level, using standard cells, and the layout can be shrunk or changed automatically, according to the technology. Specific architectures have been developed for information servers,^{4,5} and most of them use specific circuits. Recent machines deliver very high performance.⁶ Among the various accelerator types,⁶⁻¹¹ associative circuits¹²⁻¹⁴ seem to offer a natural solution for extended relational servers (see box on information servers) and for knowledge base servers.

Information servers retrieve and update data,

based on assertions of the data to be retrieved. Associative circuits also retrieve in parallel a set of data, based on a description of its content. Therefore, associative circuits seem well-adapted to the operations of extended relational servers and knowledge bases, as well as set-oriented operations on object-oriented information servers.

Associative circuits use Cartesian product algorithms, in which operations on sets of tuples (relations) are done by considering all pairs of tuples. Each cell or processing element of the associative circuit compares a tuple, or part of a tuple, which is memorized in this cell with the comparand tuple. (See box on associative circuits.) It is more efficient to join in parallel two relations by Cartesian product than to sort-join in parallel when the number of processing elements is a significant fraction of the number of tuples of one of the relations.¹⁵

The best way to satisfy this condition is to divide the operations into smaller ones compatible with the size of the associative circuits. The most efficient solutions are hash-Cartesian product algorithms. Comparisons between tuples belonging to two source relations are limited to comparisons of hashing buckets having the same index.¹⁶ Therefore, the desirable number of processors is limited to the number of tuples in the hashing bucket of the smaller relation, multiplied by the number of processing elements used for each tuple (usually one PE per tuple).

Hash-product solutions can be implemented

Information servers

Information servers manage databases in which elementary data may belong to any type and knowledge bases (databases of rules) for the benefit of users and applications distributed on an interconnection network. Database systems with the best performances for the most-used functions are extended relational systems. These systems rely on the relational data model, which defines unary and binary operations on relations. Relations are subsets of Cartesian products of domains, such as integers or real numbers. An element of a relation is a tuple, which can be represented as a record. Tuples are composed of attributes (possibly implemented as fields in a record) that take their value in the domains which define the relation.

The operations of the relational model are

- **SELECTION**, which returns the tuples that satisfy a logical formula;
- **PROJECTION**, which eliminates duplicate tuples resulting from the suppression of some attributes;
- **JOIN**, which is a selection on the Cartesian product of two relations; and
- **SET OPERATIONS** on two relations (union, intersection, and so on.)

Relational languages also include sorting, aggregates with or without grouped-by (for example, the average of a sal-

ary grouped by region), and sometimes the transitive closure on a relation. The RAPID circuit implements all these operations, though the transitive closure is not part of its instruction set.

Extended relational models define their attributes on domains corresponding to abstract data types defined by the user. They possibly can represent polygons or images. Tuples and attributes may be of any size up to several Mbytes. RAPID processes abstract data types in a simple way.

Other data models that are presently developed are object-oriented models in which methods are defined for each type of persistent or nonpersistent object and used for accessing or updating them. These object-oriented data models usually rely on object graphs in which links are represented by constant identifiers, not by pointers. These models are less advanced in terms of performance, therefore less adapted to the implementation of such servers as real-time information servers. However, they are used for an internal representation of objects in extended relational systems, which can use object managers for access and updates to objects. The software kernel of the RAPID machine uses the services of an object manager. On the other hand, object graphs can always be represented through the relation super-type, especially to execute operations on these graphs in a set-oriented manner.

by using a software hashing before the Cartesian product or by hashing the stored tuples and tuples from the second source relation by hardware at storage time and evaluation time, as it is done in some set-associative memories. In this case, however, one of the source relations must more or less remain in the associative circuit, even if the number of processing elements is smaller than the number of tuples.^{17,18}

We chose to use algorithms based on the hash-product, with a software implementation of hashing. Below, we define our circuit design goals, the data structures that the circuit recognizes, the instruction set and associated methods, and the architecture and performances. Finally, we show how this type of circuit may correspond to the needs of several generations of microprocessors.

Main issues

Today's associative circuits do not seem fully adapted to general database queries. They are either too specific or too slow. In addition, a main issue in the design of an associative circuit for databases is whether to use a small associative accelerator, which can be added to any machine, or imple-

Associative circuits

Associative circuits execute retrievals and updates on data based on

- their content,
- an assertion on this content, and, most often,
- on a comparand, or key data, which is compared to data stored into the circuit, and used to instantiate the assertion.

They are mainly composed of an array of associative cells, or processing elements, that execute comparisons in parallel, and of one or several functional blocks that use the comparison results in order to solve the query. These circuits can also execute other operations, such as sorting, proximity evaluations on the comparand and data (for example, Hamming distance), or other operations.

Associative memory capacity

Associative memories may be fully associative memories or set-associative memories. In the latter, data are distributed between sets by a hashing function, and a comparand compares to memorized data only within the set with the same hashing value.

Fully associative memories with the most advanced memory capacity offer 20 Kbits per component, which is more than 500 words with 32 bits width for each word.²² Solutions with the best integration density use five transistors per 2-bit cell, that is, 2.5 transistors per bit.²³ If used to compare tuples belonging to a couple of hashing buckets, the corresponding size of these hashing buckets is very satisfactory and is not even necessary for high efficiency. However, with this circuit type, data cannot remain in the circuit and must be transferred into it before the operation.

On the other hand, if the circuit is to be used as a data cache, it needs a bigger memory. Usual relation sizes are between 100,000 and 1 million tuples, with current tuple sizes of 100 bytes or more. Database sizes range from one Gbyte to terabytes for seismic applications or satellite data. The solution may be a set-associative memory.

With set-associative memories, we can separate logic and memory, which may be a classic RAM. We can then achieve large memory capacities, with approximatively twice the same component count and price. However, the memory capacities that are presently considered are not much larger than 10 Mbytes, which is much smaller than database sizes. The slow-down due to disk access limits the device speedup to about a factor of two (end-of-line).^{17,24} Set-associative memories have also been integrated on a single chip with memory capacities up to 160 Kbits per chip.³⁴

The hashing function depends on the operation. It may therefore require reorganizing the data before an operation, which may bring us back to the situation where data are transferred from memory. Retrieval through set-associative operations in an information server is therefore quite different from the usual use of set-associative caches, where retrieval is based on the logical addresses of data.

Due to the limited size of associative caches and needs for rehashing, smaller associative memories with non-resident data may presently be a good solution.

and frequent queries are based on joins, aggregates, sorting, and text retrieval. The three queries are implemented only by specialized associative memories.¹⁹⁻²¹ Therefore, we wanted to implement these four operations in a single associative processor. We chose to implement all query operations in the same hardware to simplify the software.

We also felt that the length of the tuple must be arbitrary. In several existing circuits, the length of a record is limited—for example, 32 bytes in Ogura's²² and 256 bytes in the Advanced Micro Devices 85C95. Larger tuples may be processed by using several components, but the record length is often a characteristic of the machine. In RAPID-1, tuples may have any length within the total memory capacity of the circuit, which may also use several components. The circuit is optimized for a useful length of the tuple (attributes taking part in the comparisons) between 4 and 100 bytes. As in Ogura's circuit and in the Data Base Accelerator,²³ the reduction of the useful tuple length may increase the number of simultaneously evaluated tuples.

Memory capacity. Considering the present limitations of associative and set-associative cache solutions, we chose to design a fully associative circuit with a small amount of memory. The general processor transfers data during all operations. Performance depends on software efficiency and on the power of the host processor. In this "co-architecture," performance increases with the power of the microprocessor, up to the saturation of the associative circuit. This arrangement is therefore well adapted to the fast evolution of general processors. (See box on associative memory capacity.)

Performance. Classic associative circuits cycle in less than 100 ns, but a comparison set implies several instructions. Comparisons using an equality criterium operate in a wide-bandwidth manner (bits in parallel mode), but comparisons on inequality criteria use serial calculations on each successive bit of the comparand (serial-bit mode). The duration of a one-word comparison is thus usually 32 cycles or more. This difference in performance between equality and inequality comparisons will imply a circuit inefficiency for inequality comparisons.

We implemented bit-parallel comparisons for all operations, which means the use of a comparator in each cell. This solution is costly in terms of circuit area, and reduces the number of tuple comparisons that can be done in parallel, but it guarantees good performance in all cases. We also used a reduced instruction set of high-level instructions, each executed in one cycle, to contribute to this purpose.

Data structures

The RAPID circuit^{25,26} recognizes two types of data structures: tuples and logical formulas. Data are memorized in the circuit as logical formulas while keys are broadcast to the processing elements as tuples. We shall explain here the

ment a complete associative machine based on an associative cache.

Functions. According to the applications, the most costly

mapping between tuples and logical formulas.

Tuples are variable-length structures, with an arbitrary number of attributes, or fields, each with an arbitrary length. The only limitation to the attribute or tuple length is the total memory size of the circuit, which may be composed of an arbitrary number of components. The internal data representation is the byte string. The comparison between numbers in two complements is mapped into a byte-string comparison by changing the appropriate bit. This operation is presently done by software. A similar approach is used for real numbers.

Each tuple is referenced by a 32-bit identifier, which may be a pointer, a link (a pointer on a pointer), or an identifier that does not change when the tuple is updated. The tuple is stored in the circuit as a logical formula, as we describe later. The tuple identifier then becomes the identifier of the corresponding subexpression. This identifier is then stored in a specific register of each cell participating in the subexpression evaluation. When the circuit compares a key tuple with a set of tuples stored as subexpressions, the result may be the sequence of identifiers of the subexpressions that are satisfied after their instantiation by the key tuple.

The circuit also recognizes logical formulas. These are composed of subexpressions that are, in turn, composed of predicates. Predicates are connected by the functions And or Or, according to the priority used in the logical formula, or by the special function Then (a variant of And), that we will explain later. The complementary function connects subexpressions. The circuit can either evaluate the global value of the logical formula or evaluate in parallel each subexpression. The result is either the set of identifiers of satisfied subexpressions or the quantity of these subexpressions. The global (Boolean) value of the formula is still available in any case. The evaluation of a fuzzy logical formula is possible, through the return of the weight of each satisfied subexpression. Such fuzzy, or vectorial, evaluation can rank documents or paragraphs by value.^{27,28} The circuit can also take into account (or not) the order of retrieved words in a sentence or document.

Predicates are either attribute-operator-constant, or attribute-operator-attribute. Each one is usually evaluated by a distinct processing element. When an argument is an attribute, it is represented by an attribute number stored in a specific location of the local memory of the processing element (PE). When a PE must compare an attribute with a constant, it waits for the arrival of the attribute number of the comparison, which usually follows the end-of-attribute (FA) signal corresponding to the previous attribute or the new-tuple (C3) signal. If the information on the data bus is equal to the predicate attribute number, the PE begins an evaluation phase. If not, the PE remains idle until the next attribute number arrives. The proportion of PEs used in parallel varies according to the query; however, in most operations it is 100 percent.

Comparison operators include $<$, \leq , $>$, \geq , $=$, \neq , Included in, Contains, Strictly included, and Strictly contains. The predicate "arg1 Included in arg2" corresponds to the search for a nonanchored byte string within another byte string. Text attributes can be divided into words that are separated by punctuation characters defined by the user. Inclusion and comparison predicates may be completed by the prefix information, which means that the retrieval is anchored at the beginning of an attribute or a word but may end before the end of the word.

There is also suffix information, corresponding to the fact that the character string does not necessarily begin at the beginning of an attribute or word, but that it ends at an attribute or word end. Specific word information indicates whether word separations are taken into account for a given predicate.

Subexpressions are defined by

subexp = predicate | subexp connect predicate, with
connect = And | Or | Then.

Then is a variant of And. It corresponds to one of the following:

- a succession of nonanchored retrievals in the same attribute;
- the comparison of tuples according to a succession of sorting criteria; or
- the successive comparison of fractions of an attribute and of a large (more than 32 bytes) constant.

In some cases, comparisons are made on calculated results. As an example, we can retrieve the names of people who earn at least 1.1 times more than the average salary of their region. (See sidebar on associative memory capacity.) In this case, the software evaluates a virtual attribute, which receives a number (for example, attribute 4), such that, Attribute 4 = $1.1 \times$ Attribute 3. This attribute is stored as a predicate constant and/or broadcast as a key in place of an ordinary attribute.

Instruction set

Figure 1 (on page 26) gives the instruction set of the RAPID-1 component. Instructions are stored in a codop (operation code) field and specified by an operator code, Oper, and by complementary bits. One instruction word is associated with each predicate, or more generally, to each PE. The circuit is controlled by data. No instruction is needed at evaluation time.

A control word (10 bits) broadcasts with every data word (16 data bits, plus 6 special bits).²⁵ Therefore, the writing of data into the circuit makes full use of the usual 32-bit word width.

SELECT
JOIN
PROJECTION
SORTING
SEMI-JOIN
SET OPERATIONS
NOP
RESERVED

Figure 1. Instruction set.

This instruction set should evolve in the next version by the suppression of the SEMI-JOIN and SET OPERATIONS instructions, which perform the same function as selection. Two important operations, EXTERNAL SORT and AGGREGATE with GROUPED BY, are executed by using the JOIN and PROJECTION operations. We briefly present the first four instructions.

SELECT. This two-phase operation loads and evaluates the selection expression. It returns a Boolean result for each evaluated tuple. The circuit overflow is not presently admitted. The SELECT query must then be decomposed into several queries.

JOIN. This is also a two-phased operation that loads tuples from one hashing bucket as a sequence of logical subexpressions, and evaluates the other hashing bucket. Predicates load sequentially. After a predicate has been stored in a PE, the PE transmits control to the next PE. For each key (which belongs to the second hashing bucket), the operation returns the sequence of tuple identifiers of the first hashing bucket that satisfies the JOIN condition. At the end of each tuple evaluation, to inform the PEs that none of them should wait to write its identifier on the bus, a specific C3 signal ends the identifiers' write phase. This signal can also interrupt the transmission of the identifier sequence.

This last mechanism is used in the EXTERNAL (distributed) SORT operation, which uses the JOIN instruction. The results of the external sort operation are the identifiers of the sorting limits. He et al. studied efficient methods for the choice of limits.²⁹

PROJECTION. The projection operation is a single load-evaluation phase. The new tuple from the current hashing bucket loads into the first free PE, and is compared at the same time with tuples loaded as a logical formula in other PEs. The representation of a set of tuples as a logical formula for a projection is given in Figure 2, where $(a(i), \dots, a(i+n))$, $(b(i), \dots, b(i+n))$, and so on, are the values of tuples that are not duplicates. If the value of the global logical formula is *false*, the current tuple is not a duplicate. In this case the software kernel stores the tuple identifier in the result relation. The control for the loading of the next tuple transfers to the next free PE. If the value of the logical formula is *true*, the

Att $i = a(i)$ and
...
Att $i+n = (i+n)$ or
Att $i = b(i)$ and
...
Att $i+n = b(i+n)$ or
...

Figure 2. A set of tuples as a logical formula.

key tuple is a duplicate. The identifier of the logical subexpression satisfied by the key tuple is returned by the circuit. This result is mainly used when using the PROJECTION operation for the calculation of an AGGREGATE with GROUPED BY. The end-of-formula token is not moved, and the PEs used to store the previous tuple are reused for the next one. This operation is fully managed by the sequencers of the relevant PEs, without using an instruction.

These mechanisms can also be used to execute AGGREGATE with GROUPED BY operations, such as the average salary per region in the *people (name, region, salary)* relation. (See box on JOIN example.) To calculate this aggregate, we build an array in which each line corresponds to a class. Columns describe the name of the class (here the region name), the intermediate calculation of the class (here the class cardinality and the total amount of salaries), then the final result for the class. Classes are created when new tuples are evaluated and correspond to new classes. Therefore the array is not ordered by class names. A sort on this column or any other may of course be done at the end of the operation; it is generally a small one. If the class is defined by an interval, the class number is calculated during a virtual attribute calculation step, before the aggregate operation itself. This solution makes possible a fast aggregate execution (due to the efficiency of hashing) and an economical one (no specific instruction). However, it increases—in a limited way—the projection cost.

INTERNAL SORT. The arguments of INTERNAL SORT are a set of tuples (small enough to fit into the accelerator) and a specific qualification expression (a list of attributes of ascending or descending sorts). The operation takes place in two phases: a load phase, in which the set of tuples is loaded into the circuit as a logical formula, and an evaluation phase, in which each tuple of the same set is successively used as a comparand. (See Figure 3.) For each comparand tuple, the associative circuit returns the tuple rank versus the other tuples of the set. The operation result is a rank array that is very comparable to that of the SORT operation in APL,³⁰ though it can include *ex aequo*, that is to say, tuples with the same rank. It is easy to use it to get a rank array without *ex aequo*, then physically reorder the tuples in sort order, but this is not

JOIN example with the RAPID circuit

Let us assume we have relations People (name, region, salary) and Region (name, average salary). The second relation may result from an aggregate operation (with a grouped by clause) on the first one. We shall retrieve the people who earn more than the average salary of their region, by executing a join between those two relations with the qualification expression

People.Region = Region.name and
People.salary \geq Region.average salary

Let us consider the following data:

People	#	name	region	salary
	1	Dupond	Paris	10
	2	Durand	Paris	20
	3	Anatole	Caen	20

Region	#	name	average salary
	1	Paris	15
	2	Caen	20

These two relations may correspond to a hashing bucket with the same number of larger relations, with a hashing function on People.Region and Region.name.

Let us assume we store the tuples from relation Region

in the circuit as logical formulas. The qualification expression will map them into the following logical formula, where subexpressions #1 and #2 will be evaluated in parallel:

PE1	ATT2 = Paris	And	#1
PE2	ATT3 \geq 15	Or	#1
PE3	ATT2 = Caen	And	#2
PE4	ATT3 \geq 20	Or	#2

Attribute numbers stored into the predicates are those of the corresponding attributes of the people relation. Virtual attribute # is the tuple identifier. Subexpression #1 corresponds to tuple #1 from region. Connector Or is mainly used as a separator between subexpressions.

In this example, tuple #1 from People, (Dupond, Paris, 10) does not satisfy the logical formula. Tuple #2 (Durand, Paris, 20) satisfies the subexpression #1 and therefore satisfies the logical formula. The result, which is then delivered by the circuit, is the following set of logical addresses: $E = \{\#1, 0\}$, where identifier 0 means "end of result." In the future, a single bit of the identifier should be used to signal the end of result while minimizing transfers. In the same way the result for tuple #3 is $E = \{\#2, 0\}$. Data broadcast to the circuit are limited to the attributes that participate in the comparison (here, attributes 2 and 3).

```

class_number = 1
While the bucket is not empty
do
  read the class_attribute of the current tuple
  store identifier = class number
  store the predicate <ATTRIBUTE = class_attribute>
  if result identifier = 0
  then begin
    store tuple identifier in result relation
    do aggregate calculus on class = class_number
      /*e.g., increment class cardinality
    class_number = class_number + 1
    move token in the PEs array
  else
    send signal C3 ("send no more identifiers")
    do aggregate calculus on class = identifier
  end
end
end

```

Figure 3. Aggregate algorithm.

always needed. It may be sufficient to access some elements of each tuple in sort order.

The tuple sets used for internal sorting are buckets resulting from a distributive sort. From the rank array of each bucket, it is usually necessary to calculate the ranks array versus the whole relation. This is done by adding a constant to the ranks of each bucket.

During the loading phase of each sort bucket, sort attributes are stored in the circuit as predicates (for example, as "attribute < constant" in the case of an ascending sort attribute). Successive sort predicates are connected by a Then connector. If successive sort attributes correspond to the same sorting sense, it is possible to compact them in the same PE, in the limits of the memory capacity of the PE. During the evaluation phase, the rank calculation is a count of the number of satisfied subexpressions. In the case of an ascending sort, the number of satisfied expressions would be the number of inferior tuples in sort order. He et al. simulated parameters for an optimization of external and internal sort using the RAPID circuit.²⁹

Architecture

The RAPID accelerator (Figure 4) is an associative circuit that interfaces with the host machine as a memory. Figure 5 shows it is composed of three main functional blocks: the array of associative cells or PEs, the subexpressions resolution block, and the query resolution block. In contrast with other associative circuits such as Ogura's and the Data Base Accelerator, it does not have a bit-management block, because it does not execute bit serial operations. The subexpression resolution block is a specific logical block that executes Ands, Ors, or Thens between predicates. Its operation is detailed later. Most associative circuits do not include this type of functional block, but Ogura et al.²² proposed a functionally similar structure, and Penazola and Ozkarahan³¹ proposed a seemingly similar one. The query resolution module includes the And block, the Or block, the counting block ("+" block), and the arbitration block.

The arbitration block is the classic multiple-match resolution circuit, which is a characteristic of associative memories. (See Figure 6.) As this block is mainly implemented using a single, programmable component, and the length of the circuit in gates is a logarithm of the number of components, its delays are not very dependent on the number of PEs or of components, which is usually between eight and 16.

The And block calculates a logical And between ends of subexpression values, which enables an evaluation of logical formulas in Or priority. The Or block executes a logical Or between the same values, which is used for the evaluation of logical formulas in And priority. As this representation normally represents tuples as a logical formula, the Or block can also check if one tuple at least satisfies the comparison condition with the key tuple. The counting block counts the number of true subexpressions, which is used for the internal sort operation.

Processing elements are all at the

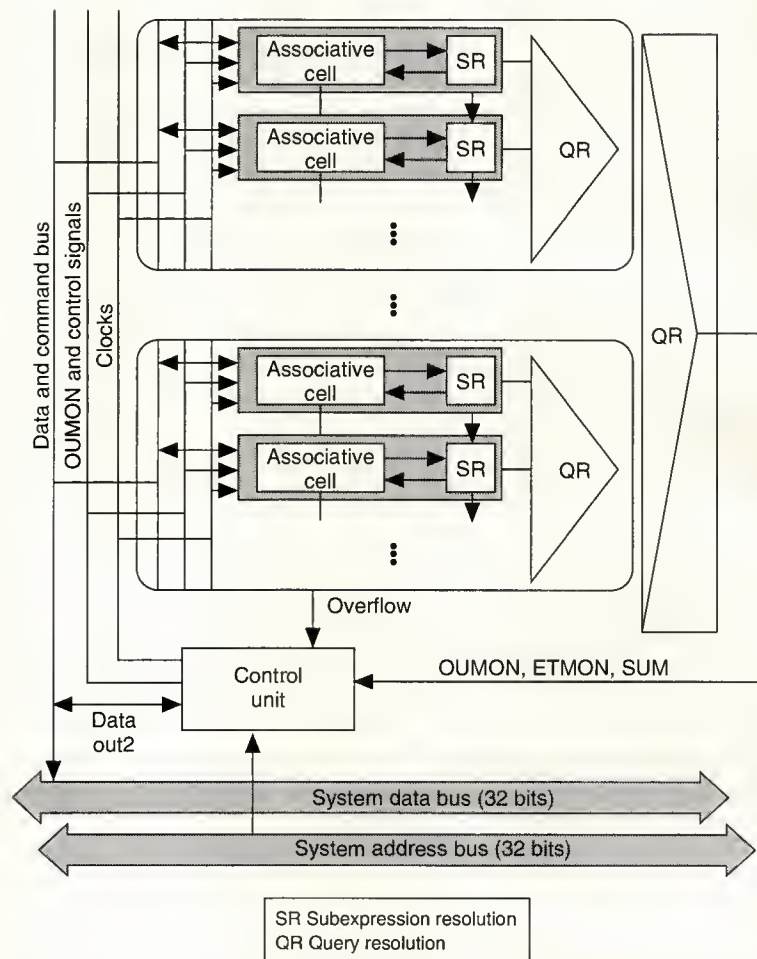


Figure 4. RAPID board.

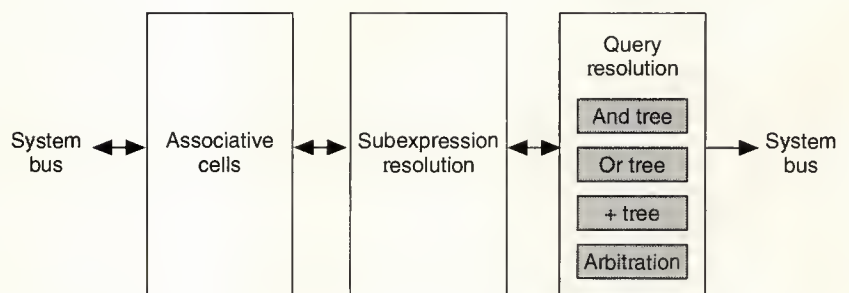


Figure 5. Block diagram of the accelerator.

same address for writes and reads. Writes and reads are done according to a protocol. Data control the circuit. Management of the communication protocol with the host machine bus is at the board level, to make the circuit independent from a given host machine. The board usually produces a clock cycle only if the host processor has written or read data in the circuit at the end of the previous cycle. However, the circuit is synchronized with the host machine.

The associative circuit may be implemented into one or several main components, each including several PEs and part of the other functional blocks. With more than a few tens of PEs per machine, the performance of the accelerator becomes independent from its number of PEs, except for very large relations (more than one million tuples). The subexpression resolution block is fully distributed on the main components, based on one resolution module per PE. Each resolution module cascades with both the previous and next ones. The part of the query resolution block common to several main components is presently constructed with off-the-shelf components.

Processing element. The PE is the basic cell of RAPID. It executes the embedded operations and memorizes data. The external view of the PE in Figure 7 displays its interconnections with its environment. These connections include the internal data bus, command bus, control signals, connections with the previous and next PE, and pins signaling the relative position of the PE in the PE array. Table 1 on page 30 lists descriptions of the corresponding signals.

An internal view of the PE (Figure 8 on page 31) reveals an operative part and a control part. The operative part includes a local memory with 18 words for the storage of a 16-word operand and two attribute numbers. Each word is 16 bits wide, and an attribute number also has 16 bits. The operative part also includes a logical comparator, a predicate generator for PRED values, and several registers. The registers include two for the external and internal configuration of the PE (STATUS1, which holds the instruction word and is read/write, and STATUS2, which holds the PE state and is read-only), two for the interface with the PE environment (INOUT and MCR), and a fifth for the storage of the tuple identifier (IDENT).

The control part is the PE sequencer, which is divided into

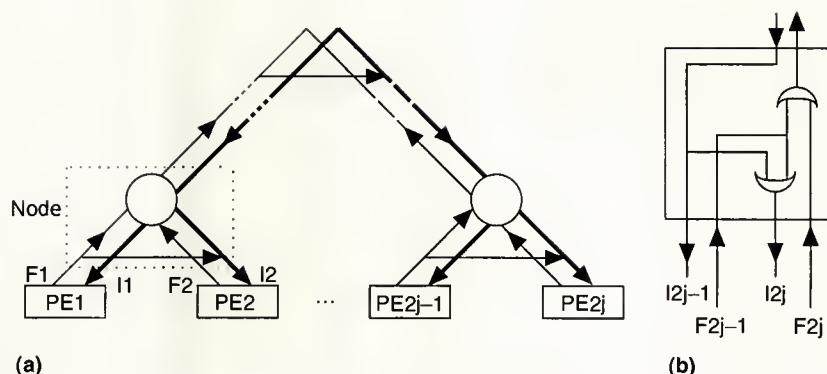


Figure 6. Arbitration block, from Anderson³² (a); close-up of node (b).

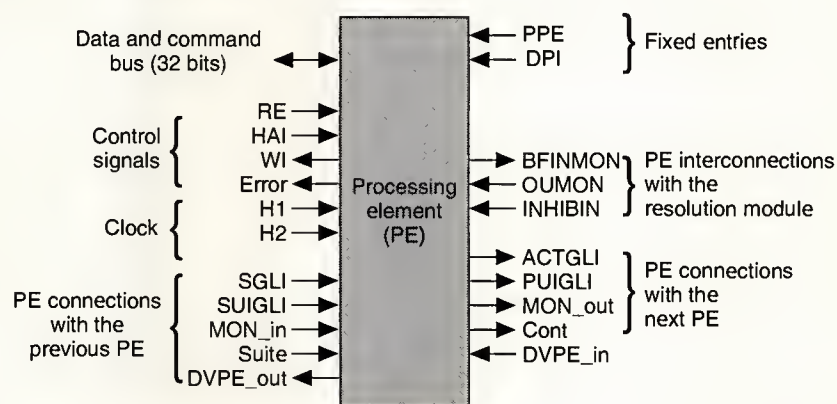


Figure 7. External connections of the PE.

three automata. The test automaton (AUTOT) enables a very fast access to the operand, configuration, and internal registers during normal operation. The primary automaton (AUTOP) schedules the major part of the PE's operations. The auxiliary automaton (AUTOA) helps the main automaton during the loading phase.

Subexpressions resolution. The subexpressions resolution (Figure 9 on page 31) implements a chained And, a chained Or, and a chained Then. MON_in and MON_out are the subexpression values, CONX the connector value, PRED the predicate value, and FINMON the result. We have detailed the subexpression in earlier papers.^{25,26} The response time of the subexpressions resolution circuit equals the propagation time in the logical groups corresponding to the maximum size of a subexpression. We can determine the response time by considering the propagation time on all modules, or by choosing a maximum size for a monomial or subexpression.

An important problem in the circuit design is the processing of large predicates, in which the constant overflows the

Table 1. External PE signals.

Command signals
C1: starting a loading phase at next cycle C2: starting an evaluation phase at next cycle C3: broadcast of a tuple to be evaluated at the next cycle FA: end of an attribute FR: end of the relation FTUP: end of a tuple FMOT: end of a word in a text ADS: presence of a PE address on the data bus CTRL0/T1: opening parenthesis in the normal mode//a bit of the test instruction code in the test mode CTRL1/T2: closing parenthesis in the normal mode//a bit of the test instruction code in the test mode
PE's chain signals
MON_in/MON_out: partial subexpression value propagation Suite/Cont: control transmitted from one PE to another to make it the current PE SGLI/ACTIGLI: activation of the PEs that contains the continuation of a nonanchored search by one PE SUIGLI/PUIGLI: activation of the PE group containing the continuation of a nonanchored search by a group of PEs DVPE_in/DVPE_out: propagation of a tuple overflow signal in a loading phase
Other signals
OUMON: input pin, receives the result of the Or resolution tree RE: PE initialization input HAI: PE entry on the PAUSE mode (especially on the test mode) WI: output pin, bus capture by the PE Error: output pin, error detection by the PE PPE: fixed pin, used to distinguish the first PE of the PEs vector DPE: fixed pin, used to distinguish the last PE the PEs vector

local memory of one PE. Several solutions have been proposed for that problem, including an And of comparisons on several operand parts identified by an index,²² the addressing of a group of cells corresponding to an address where low-weight bits are masked,²³ or the transfer of a control token from one byte to the next in ISSP.^{20,21} We pass control from

one PE to the next when the comparison on the current operand part ends, and when the comparison result is equality. The corresponding predicate value is then forced to the neutral value for the subexpression resolution, true-in-And priority. The only significant value of PRED is that of the first PE that returns an inequality comparison or the value of PRED in the last PE. The PRED values for the next PEs in the same predicate—if any exist—remain at the neutral value.²⁵

Rapid versions. We designed version V0 to prove the circuit feasibility. It implements all relational operations extended to sorting. It was customized using a Silvar Lisco CAD software. A component contains a single PE, in a 68-pin CLCC package. Fifteen of the pins are free for further versions. Table 2 gives the characteristics of this version.

With a host machine powerful enough to saturate the accelerator, V0 executes a $1,000 \times 1,000$ -tuples JOIN in 4 ms. However, the limitation of the total number of PEs on an actual board implies an important overhead in the case of large relations.

Implementation of V0 effectively began in 1988, but stopped in 1989 due to the departure of the logic designer. At this time, we had only a partial simulation. Checks at logic and layout levels were resumed in spring 1990 by a PhD student. The ES2 Company fabricated it in October 1990.

We designed most of V1 from scratch. V1.1 is compatible with the SQL-1 standard and implements four PEs on each component. A practical board with eight main components may then include 32 PEs, enough for current performance standards. We designed this version with standard cells and a 1.5-micron technology, using the Cadence CAD package.

V1.1 includes text retrieval, detection, and processing of null values according to the SQL-1 standard and the processing of overflows. Modifications of the PROJECTION operation enable it to use the AGGREGATE with GROUPED BY function, according to the method described earlier. We also improved testability. We added scan paths, which make it possible to separately scan four parts of the circuit. Using the test automaton, we can rapidly access the internal states of the circuit during normal operation and therefore run fast test programs. The effective version design started in summer 1990 with one PhD student, and we expect the circuit to be fabricated by the end of 1991.

Environment and performance

With the RAPID accelerator, access to data, hashing, data transfer to the accelerator, and results transfer into result relations, are done by a software kernel executed by the host processor. This kernel is composed of a multiuser transactions manager that manages threads (lightweight processes), a query machine that executes query operations, and an object manager that does object creation and access. The thread management by the software kernel guarantees a low-cost, atomic execution of operations by the accelerator, that

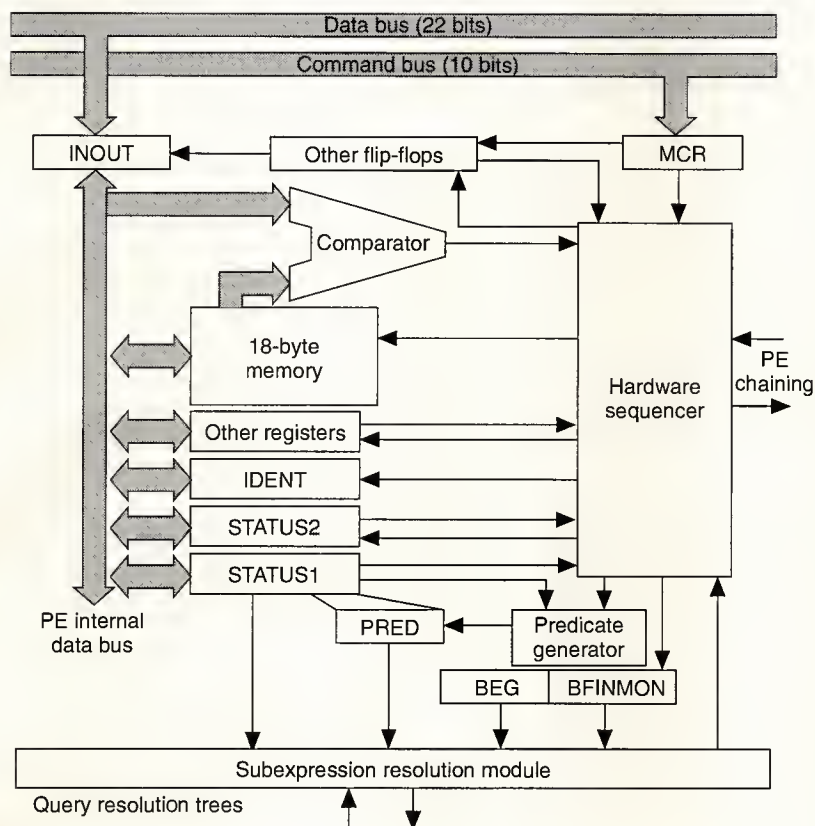


Figure 8. Internal view of the processing element.

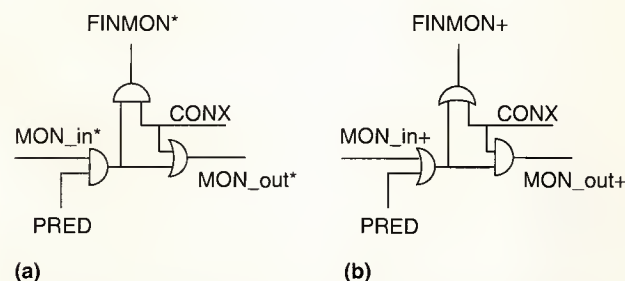


Figure 9. Subexpressions resolution: And priority (a); Or priority (b).

is to say, an execution in which a suboperation is never pre-empted. This atomicity is needed for performance reasons. The accelerator executes the operations. However, in the case of selections, preselections of the most selective attribute are done using an index. Software kernel performance is a

major factor of RAPID's efficiency.

We developed the first version of the software kernel on a Sun 3/50, with a recent object manager. We measured the duration of the 1,000 × 1,000-JOIN (a JOIN of 1,000 tuples per 1,000 tuples returning 1,000 result tuples) in the software configuration that corresponds to the use of the accelerator. We have not used the accelerator board in the Sun workstation. (One version of the board has been implemented in a Macintosh II.) The software operation time is 1.23 seconds. The product of this time and the number of MIPS of the machine (about 2.5) is better than the throughput of the Gamma machine, which is a well-known parallel machine that uses about 7 MIPS to execute this operation in memory.³³ The time lapse in the accelerator is very small, about 4 ms with the first version of the circuit.

We are improving the performance with a novel, performance-oriented, object manager. We have completed the design of this object manager and estimate it will reduce the access time by approximately one order of magnitude in an average case. This object manager is a critical module. By using it and some complementary software improvements, the software duration of the reference operation should drop to about 200 ms, while the time spent in the hardware should be about 2.58 ms with the V1 version.

Figure 10 on page 32 shows the improvement ratio at a given number of MIPS, based

Table 2. Features of RAPID V0.

Configuration	Number of PEs × 16 words × 22 bits
Instruction set	7 instructions
Cycle time	120 ns
Supply voltage	5 V
VLSI process technology	2-μm CMOS with double aluminium layers
Number of devices	11,000
Chip size	5.3mm × 5.2mm
Number of pins	50
Package	68-pin CLCC

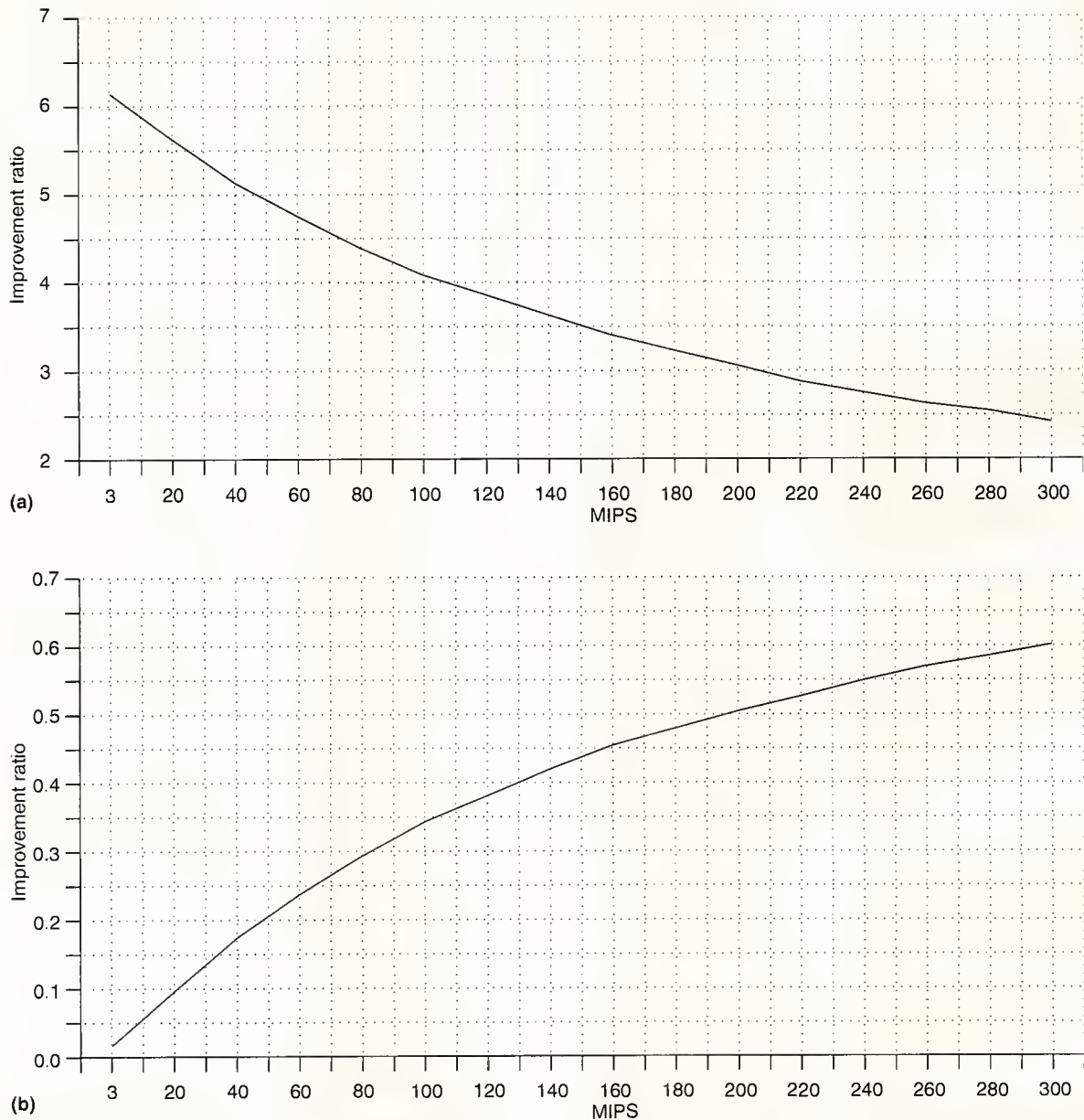


Figure 10. Improvement ratio vs. host processing power (a); accelerator utilization ratio vs. host processing power (b).

on our assumptions. We assume the host machine throughput in Mbytes equals the number of MIPS. But the throughput needed for the circuit I/Os is much smaller, about 10^5 bytes/s \times MIPS. Clearly, the accelerator can significantly improve performance even on a 200-MIPS host machine.

THE RAPID-1 IS AN ASSOCIATIVE ACCELERATOR for database operations. Its instruction set includes relational operations extended to SORT and AGGREGATE with GROUPED BY. Data do not remain in the associative circuit, except for very short durations. Programming is a mere trans-

lation of query predicates, though an adequate software kernel is needed for efficiency.

The circuit uses a small number of associative cells with hardwired control. It speeds up database systems by a factor of five, versus most referenced database systems. The acceleration ratio will still be very significant with the emerging class of new microprocessors, which may have a peak power about or above 100 MIPS. The accelerator will also speed up parallel database systems using these processors.

These results show that the choice of an associative circuit with hardwired control, that is controlled by data, with bit-parallel comparisons for all operations, is critical for the use of the accelerator with new generations of processors. Other solutions are possible but would lead to machines with weaker performance (due to multiple-instruction transfers) and less homogeneity when inequality predicates are used (such as for sorting).

We hope to see this accelerator evolve further to include the processing of binary strings, fault tolerance, a feasible performance increase by a factor of 10 to meet the needs of the next generation of microprocessors, and the design of components with more parallelism. ■

Acknowledgments

We thank all the members of the RAPID team for their very fruitful contribution and especially E. Abecassis, D. Donsez, G. Fouquet, D. Guidot, H. He, and J. Penne. We also thank the anonymous referees for fruitful comments resulting in several improvements of this article.

The MASI Laboratory is an associated unit of the Centre National de la Recherche Scientifique. The RAPID project has been partly financed by Agence Nationale de Valorisation de la Recherche (ANVAR), and is now partly financed by the French Ministère de la Recherche et de la Technologie (AASI-89 program), and by the PRC-AMN, a French Cooperative Research Program in Novel Machine Architectures at the Centre National de la Recherche Scientifique.

References

1. H. Boral and D.J. Dewitt, "Database Machines: An Idea Whose Time Has Passed?" *Proc. Third Int'l Workshop Database Machines*, Springer-Verlag, New York, 1983, pp. 167-187.
2. Laguna Beach Group, "The Laguna Beach Report: Future Directions in DBMS Research," *SIGMOD Record*, Vol. 18, No. 1, Mar. 1989.
3. G. Musgrave, ed., *Proc. VLSI*, International Federation for Information Processing, Vol. 10.S, Munich, Aug. 1989.
4. *DBC/1012 Data Base Computer Concept and Facilities*, Document No. C02-000100, Teradata Corp., 1983.
5. Tandem Performance Group, "A Benchmark of Non-Stop SQL on Debit-Credit Transaction," *Proc. ACM-SIGMOD*, ACM, N.Y., 1988, pp. 337-341.
6. M. Kitsuregawa, "Implementation of LSI Sort Chip for Bimodal Sort Memory," *Proc. VLSI*, International Federation for Information Processing, 10.S, 1989, pp. 285-294.
7. M. Abdelguerfi and A.K. Sood, "A Bus-Connected Cellular Array Processing Unit for Relational Database Machines," *Proc. Fifth Int'l Workshop Database Machines*, Kluwer Academic Publishers, Boston, 1987, pp. 188-201.
8. K.C. Lee and G. Herman, "A High-Performance VLSI Data Filter," *Proc. Fifth Int'l Workshop Database Machines*, Kluwer Academic Publishers, 1987, pp. 251-268.
9. D.J. DeWitt and H. Boral, eds., "Database Machines," *Proc. Fourth Int'l Workshop Database Machines*, Springer-Verlag, 1985.
10. M. Kitsuregawa, ed., *Proc. Fifth Int'l Workshop Database Machines*, Kluwer Academic Publishers, 1987.
11. H. Boral and P. Faudemay, "Database Machines," *Proc. Sixth Int'l Workshop Database Machines*, Lecture Notes in Computer Science, No. 368, Springer-Verlag, 1989.
12. J. Minker, "An Overview of Associative and Content-Addressable Memory Systems and a KWIC Index to the Literature," *Computing Reviews*, Oct. 1971.
13. K.J. Thurber and L.D. Wald, "Associative and Parallel Processors," *Computing Surveys*, Vol. 7, No. 4, Dec. 1975, pp. 215-256.
14. L. Chisvin and R.J. Duckworth, "Content-Addressable and Associative Memory: Alternatives to the Ubiquitous RAM," *Computer*, Vol. 22, No. 7, July 1989, pp. 51-64.
15. D. Bitton et al., "Parallel Algorithms for the Execution of Relational Parallel Operations," *Trans. Database Systems*, Vol. 8, No. 3, Sept. 1983, pp. 324-353.
16. M. Kitsuregawa et al., "Application of Hash to Database Machine and its Architecture," *New Generation Computing*, Vol. 1, No. 1, 1983, pp. 63-74.
17. S.H. Lavington and R.A.J. Davies, "Active Memory for Managing Persistent Objects," *Proc. Int'l Workshop on Computer Architectures to Support Security and Persistence*, Springer-Verlag, May 1990, pp. 137-154.
18. S.H. Lavington and J. Robinson, "Exploiting Data Parallelism in Knowledge-Based Systems," Research Report CSM-158, Department of Computer Science, Essex University, U.K., 1990.
19. M. Abdelguerfi, "Special Function Unit for Statistical Aggregation Functions," *Proc. Sixth Int'l Workshop on Database Machines*, Lecture Notes in Computer Science, No. 368, Springer-Verlag, 1989, pp. 187-201.
20. K. Takahashi, H. Yamada, and M. Hirata, "A String Search Processor LSI," *J. Information Processing*, Vol. 13, No. 2, 1990, pp. 183-189.
21. K. Takahashi et al., "A New String Search Hardware Architecture for VLSI," *Proc. 13th Int'l Symp. Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., June 1986, pp. 20-27.
22. T. Ogura et al., "A 20-Kbit Associative Memory LSI for Artificial Intelligence Machines," *IEEE J. Solid-State Circuits*, Vol. 24, No.

- 4, Aug. 1989, pp. 1014-1027.
23. J.P. Wade and C.G. Sodini, "A Ternary Content Addressable Search Engine," *IEEE J. Solid-State Circuits*, Vol. 24, No. 4, Aug. 1989, pp. 1003-1013.
24. S.H. Lavington et al., "Hardware Memory Management for Large Knowledge Bases," *Proc. PARLE (Parallel Architectures and Languages Europe)*, Vol. I, Lecture Notes on Computer Science, No. 258, Springer-Verlag, 1987.
25. P. Faudemay et al., "The Database Processor RAPID," *Proc. Fifth Int'l Workshop on Database Machines*, Kluwer Academic Publishers, 1987, pp. 171-187.
26. P. Faudemay, *Un Processeur VLSI pour les Operations de Bases de Donnees* (A VLSI Processor for Database Operations), PhD thesis, University Pierre et Marie Curie, Paris, 1986.
27. G. Salton, E. Fox and H. Wu, "Extended Boolean Information Retrieval," *Comm. ACM*, Vol. 26, No. 12, Dec. 1983, 1022-1036.
28. C. Stanfill and B. Kahle, "Parallel Free-Text Search on the Connection Machine," *Comm. ACM*, Vol. 29, No. 12, Dec. 1986, pp. 1213-1228.
29. H. He, P. Faudemay, and L. Chen, "Le tri, la projection et les agregats dans la machine bases de donnees RAPID," ("Sorting, Projection, and Aggregates in the RAPID Database Machine."), research report, Institut Blaise Pascal, 1990.
30. K. Iverson, *A Programming Language*, John Wiley & Sons, N.Y., 1962.
31. M. Penazola and E. Ozkarahan, "Integrating Integrity Constraints With Database Filters Implemented in Hardware," *Proc. Sixth Int'l Workshop on Database Machines*, Lecture Notes in Computer Science, No. 368, Springer-Verlag, 1989, pp. 230-250.
32. G.A. Anderson, "Multiple Match Resolvers: A New Design Method," *IEEE Trans. Computers*, Vol C-23, No. 12, Dec. 1974, pp. 1317-1322.
33. D.A. Schneider and D.J. DeWitt, "A Performance Evaluation of Four Parallel Join Algorithms in a Shared-Nothing Multiprocessor Environment," *Proc. ACM-SIGMOD*, ACM, 1989, pp. 110-121.
34. M. Motomura et al., "A 1.2-Million-Transistor, 33-MHz, 20-b Dictionary Search Processor (DISP) ULSI with a 160-Kbyte CAM," *IEEE J. Solid-State Circuits*, Vol. 25, No. 5, Oct. 1990, p. 1158-1165.



Pascal Faudemay is project manager of the RAPID project at MASI Laboratory and an engineer at the Centre National de la Recherche Scientifique. He designs performance-oriented software and hardware architectures for databases. His interests include microarchitecture, databases, object servers, real-time processing, and parallel and distributed systems. Faudemay earned BS and MS degrees from University Paris I and a PhD in computer science from University Pierre et Marie Curie in 1986. He is a member of the IEEE Computer Society and ACM.



Mongia Mhiri is a PhD student in computer science at University Pierre et Marie Curie. She works on version V1 of the RAPID associative circuit at MASI Laboratory. Mhiri received BS and MS degrees in computer science from Grenoble University in 1989.

Address questions concerning this article to Pascal Faudemay, Laboratoire MASI, University Pierre et Marie Curie, 4 Place Jussieu, 75252 Paris cedex 05, France; or via e-mail at faudemay@masi.ibp.fr.

Expedited delivery

is available to all members residing outside the USA, Canada, and Mexico. We invite you to take advantage of this service providing delivery of your magazine weeks earlier.

For information on this service and its cost, contact:

**Expedited Delivery
IEEE Computer Society
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264 USA
Phone: (714) 821-8380
FAX: (714) 821-4010**

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 156

Medium 157

High 158



A Fine-Grain Architecture for Relational Database Aggregation Operations

This design and simulation of a bit-sliced processor for relational database aggregation functions addresses an important, computationally expensive problem in database computers. The slice processor takes two tuples as inputs (one bit at a time) and returns two bits as outputs every clock cycle. A larger aggregation unit uses a number of identical slice processors, connected according to odd-even network topology, to achieve improved performance on a parallel pipelined processor.

M. Abdelguerfi

University of New Orleans

A.K. Sood

George Mason University

Since its introduction by Codd in 1970,¹ the relational database model has achieved wide acceptance. While allowing a high degree of data independence, the model provides a theoretical means to deal with consistency and redundancy problems. This is possible mainly because data is represented in two-dimensional tables that can be designed using the normalization theory.

In the relational database model, data can be manipulated through a small but powerful set of operators. These operators are the relational algebra operations (such as PROJECTION, JOIN, and SET) and aggregation operations. Most of these operations are computationally intensive and algorithm design for efficient implementation plays an important role in the design of database computers. Efficient processing has a determining effect on system performance. As a result, the operations have been the subject of intense study in the development of relational database management systems.

For instance, to maximize the performance of the natural JOIN operation, several algorithms have been developed.² While these methods tended to be effective in conventional von

Neumann computer systems, a closer look reveals great opportunity for concurrent processing. With appropriate hardware and system organization, such concepts as parallelism and pipelining can significantly enhance the performance of these operations.

We designed and simulated a special-purpose unit for such aggregation functions as Sum, Count, and Average using a one-bit slice processor as a basic component. (We do not consider the remaining two statistical aggregation functions Min and Max here as they are not computationally intensive.) A larger unit can be easily designed by connecting several identical slice processors according to odd-even network topology.³ In this unit the tuples are input, processed, and output in parallel, one tuple at a time. The processing elements operate on tuples bit by bit. As a result, we refer to this system as parallel bit-level pipelined architecture.⁴

The main advantage of this approach is that the memory requirement of each slice processor is very small and is independent of input size. Since input operands process one bit at a time, we reduce the amount of hardware in each slice processor. As a result, a large number of slice

processors can be integrated on one VLSI chip. The proposed design exhibits several attractive features.

- **A simple, basic slice processor.** The system makes use of only one type of simple slice processor. Each slice operates on data bit by bit, simplifying design and verification of the circuit.
- **Overlap of data I/O and processing.** Data processing time is completely overlapped with data input and output to and from each slice.
- **Static interconnection network.** For ease of implementation, the design uses a static interconnection between the different slices rather than a dynamic network. This connection allows the system to process several data streams in a pipelined manner.
- **High throughput.** Several bit-serial slices operating in parallel achieve high throughput.
- **Low pin count.** The input and output of operands one bit at a time, to and from the slices, ensure a low count.
- **Design is independent of tuple size.**
- **Limited interconnection requirement.** Bit-serial computation limits interconnection needs.

Output of the tuples is completely overlapped with their input and processing times. Therefore, a tuple stream input to the unit processes in a pipelined way, one-bit per tuple at a time. Several different input streams of tuples can also be processed in a pipelined manner. That is, the processing of a new input stream of tuples can be initiated while the previous stream is still being processed. Consequently, the proposed unit achieves pipelining at both the tuple and input stream levels.

We use a SIMD (single-instruction stream, multiple-data) architecture to perform statistical aggregation functions. In contrast to the parallel pipelined query processor approach in Kim et al.,⁵ sorting, aggregation, and duplicate marking are performed concurrently. One network topology and one type of processing element (slice) implement these functions. The design in Kim et al. uses three different network topologies and processes parallel tuples one bit at a time.

The processor

Let's examine the design of our Sum processing unit. The same unit will be used to perform the COUNT and AVERAGE operations.

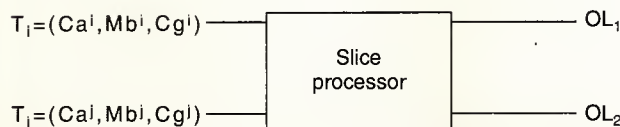


Figure 1. A slice processor.

```

Procedure SUM ( (Cai,Mbi,Cgi), (Caj,Mbj,Cgj) )
/* Two reduced tuples Ti = (Cai,Mbi,Cgi) and Tj = (Caj,Mbj,Cgj)
are input to a slice */
begin
  if Cgi > Cgj then
    begin OL1 := (Cai,Mbi,Cgi) ; OL2 := (Caj,Mbj,Cgj) ;
  elseif Cgi < Cgj then
    begin OL1 := (Caj,Mbj,Cgj) ; OL2 := (Cai,Mbi,Cgi) ;
  else
    if Mbi AND Mbj = 1 then
      OL1 := (Cai + Caj,Mbi,Cgi) ; OL2 := (-,0,Cgj) ;
    elseif Mbi AND Mbj = 1 then
      OL1 := (Caj,Mbj,Cgj) ; OL2 := (-,Mbi,Cgi) ;
    else
      OL1 := (Cai,Mbi,Cgi) ; OL2 := (Caj,Mbj,Cgj) ;
    endif ;
  endif ;
end;

```

Figure 2. Procedure Sum algorithm.

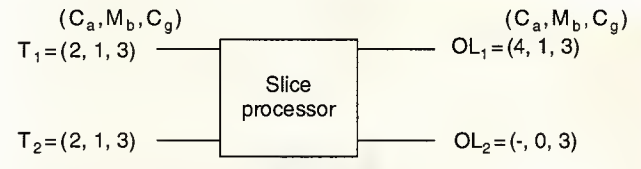


Figure 3. Example of a Sum algorithm.

Methodology. To each reduced tuple, we associate a mark bit we call Mb. Each one-bit slice processor takes as inputs two reduced tuples (see Figure 1). The slice processor compares the two Group-By values (Cgⁱ, Cg^j). It manipulates the aggregation values (Caⁱ, Ca^j) and mark bits (Mbⁱ, Mb^j) according to the result of the comparison phase. The resulting reduced tuples are output through OL₁ and OL₂. The mark bits identify the qualified tuples, which are output with a mark bit set equal to 1. Thus, the processor filters out duplicate tuples by examining their mark bit. Figure 2 shows the algorithm performed by the slice processor.

An example appears in Figure 3 in which two tuples (T₁ and T₂) are input to the slice processor. These tuples are both qualified (their mark bits are both 1) and have the same Group-By value. Therefore, their aggregation columns will be summed, and one of the tuples will be disqualified by resetting its mark bit to 0.

Hardware implementation. Figure 4 depicts a block diagram of a one-bit slice processor that contains five flags: F₁, F₂, F₃, F₄, F₅. The two reduced tuples are input, processed, and output one bit at a time. Processing starts with the Group-By values Cg^m = {Cgk^m, Cg2^m, ..., Cgl^m} m = i,j, which are

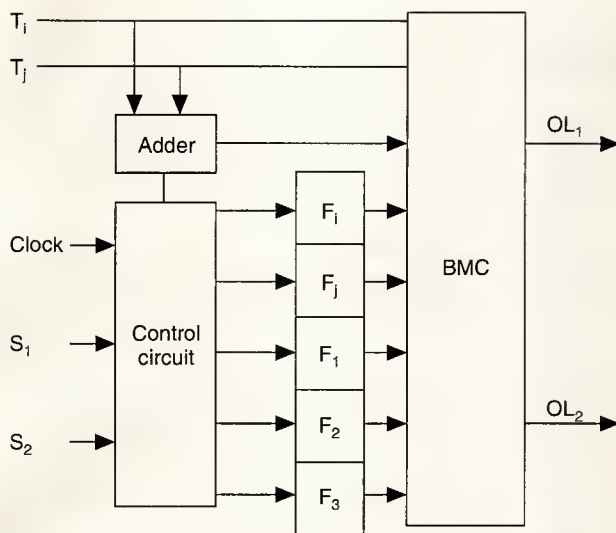


Figure 4. Block diagram of a slice processor.

input to the slice processor bit by bit starting from the most significant bits, or MSBs, (Cgk^i , Cgk^j). (Here, k represents the number of bits in the binary representation of the Group-By values.) This makes the bit-serial comparison of the two Group-By values easier to perform.

Next, mark bits Mb^i and Mb^j are input, followed by the aggregate values $Ca^m = \{Ca1^m, Ca2^m, \dots, Cap^m\}$, $m = i, j$. Aggregate values must be added serially, bit by bit, starting from the least significant bits, or LSBs. Therefore, unlike the Group-By values, the aggregate values are input starting from the LSBs ($Ca1^i$, $Ca1^j$). The number of bits in the binary representation of the aggregate values is p . Notice that the Group-By values comparison, the manipulation of mark bits, and the summation of aggregate values completely overlap with the input and output of the reduced tuples to and from the slice processor.

Control and synchronization are achieved through two control signals S_1 and S_2 . S_1 , a start signal, is applied to the slice processor at time instant t_0 for one clock cycle. During this time instant, the slice processor resets its flags. At the next time instant, the slice processor begins the processing of the reduced tuples present at its two inputs.

Control signal S_2 indicates the completion of Group-By values comparison. This signal is applied to the slice processor at time instant t_k for one clock cycle. At the same time instant, the two mark bits are input to the slice processor. From time instant t_{k+1} to t_{k+p} , the aggregate values are input to the slice processor.

The synchronous serial adder computes the sum ($Sum = Ca^i + Ca^j$). CF denotes the adder carry flag (not shown in the block diagram). Flags F_i and F_j store mark bits Mb^i and Mb^j . Control signal S_2 sets flag F_3 , which indicates that Group-By values

comparison has been completed. Finally, flags F_1 and F_2 store the outcome of the comparison according to Table 1.

The Bit Manipulation Circuit (BMC) determines the two outputs $OL_m = (Cgk^m, Cg2^m, \dots, Cg1^m, Mb^m, Ca1^m, Ca2^m, \dots, Cap^m)$ and $m = 1, 2$. The Bit Serial Sum algorithm (see the box) computes each output bit. The slice processor starts the output process one clock cycle after the input process begins. Thus, the output process completely overlaps both the input and processing times. Note that the hardware algorithm is independent of the tuple size. Because tuple size varies from one application to another, this characteristic is crucial. As shown in Figure 5, by controlling the time interval between control signals S_1 and S_2 , the slice processor can be tuned to a particular tuple size.

We designed a slice processor using 10 D flip-flops and about 72 gates. Figures 6 and 7 show an example (with $p = 2$ and $k = 4$) and its simulation. In this example, the simulation starts at the trailing edge of the first appearance of S_1 . The output process starts one cycle after the input process begins.

Implementing other aggregation operations. The Count and Average functions can be implemented using the Sum unit. For the COUNT operation, each reduced tuple will be input to the Sum unit with a Ca value set equal to 1. Query

Table 1. Flags F_1 and F_2 .		
F_1	F_2	Condition
0	0	$Cg^i = Cg^j$
0	1	—
1	0	$Cg^i > Cg^j$
1	1	$Cg^i < Cg^j$

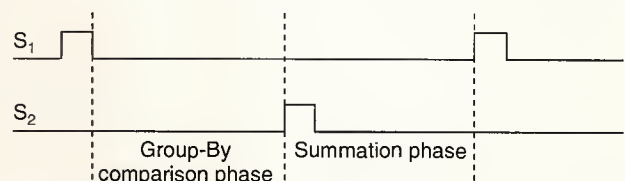


Figure 5. Timing diagram.

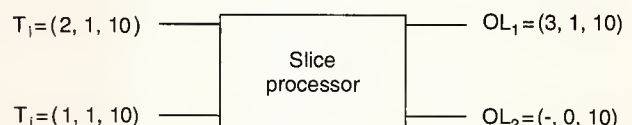


Figure 6. Example of a one-slice processor operation.

The Bit-Serial Sum Algorithm

Procedure Bit-Serial SUM (T_i, T_j)

/* Two tuples $T_i = (Cgk^i, \dots, Cg2^i Cg1^i Mb^i Ca1^i Ca2^i, \dots, Cap^i)$ and $T_j = (Cgk^j, \dots, Cg2^j Cg1^j Mb^j Ca1^j Ca2^j, \dots, Cap^j)$ are processed one bit at a time by the slice processor. The outputting of the result is completely overlapped with the inputting process. The slice processor two outputs are $OL_m = (Cgk^m, \dots, Cg2^m Cg1^m Mb^m Ca1^m Ca2^m, \dots, Cap^m)$, $m=1,2$ */
if $S_1 = \text{true}$ then begin /* Initialize flags (the duration of S_1 is one clock cycle)*/

$F1 := 0$; $F2 := 0$; $F3 := 0$; $Fi := 0$; $Fj := 0$; $n := k$;

while $S_2 = \text{false}$ and $n \geq 1$ loop /* Start the GROUP BY comparison phase */

if $(C_{gn}^i > C_{gn}^j)$ then

if $(F1 = 0)$ then $C_{gn}^1 := C_{gn}^i$; $C_{gn}^2 := C_{gn}^j$; $F1 := 1$;

elseif $(F2 = 0)$ then $C_{gn}^1 = C_{gn}^i$; $C_{gn}^2 := C_{gn}^j$;

else $C_{gn}^1 := C_{gn}^j$; $C_{gn}^2 := C_{gn}^i$;

endif;

elseif $(C_{gn}^i < C_{gn}^j)$ then

if $(F1 = 0)$ then $C_{gn}^1 := C_{gn}^j$; $C_{gn}^2 := C_{gn}^i$; $F1 := 1$;

$F2 := 1$;

elseif $(F2 = 0)$ then $C_{gn}^1 := C_{gn}^j$; $C_{gn}^2 := C_{gn}^i$;

else $C_{gn}^1 := C_{gn}^i$; $C_{gn}^2 := C_{gn}^j$;

endif;

else $C_{gn}^1 := C_{gn}^i$; $C_{gn}^2 := C_{gn}^j$;

endif;

$n := n - 1$;

end;

/* S_2 is now high for one clock cycle. During this Phase (called mark bit manipulation phase) the two mark bits are stored (in Fi and Fj) and a mark bit manipulation is performed. The phase duration is one clock cycle*/

$F_i := M_b^i$; $F_j := M_b^j$; $CF := 0$;

if $(F1 = 0)$ then $M_b^1 := (M_b^i \text{ or } M_b^j)$; $M_b^2 := 0$;

else $M_b^1 := M_b^i$; $M_b^2 := M_b^j$;

endif;

$m := 1$

/* Now the summation phase begins. This phase will last until a new start signal (S_1) is applied to the processor*/

while $S_1 = \text{false}$ and $m \leq p$ loop

if $(F1 = 0)$ and $(Fi = 1)$ and $Fj = 1$ then /* "-" refers to don't care*/

$Cam^1 := Cam^i \text{ XOR } Cam^j \text{ XOR } CF$; $Cam^2 := -$; CF

$:= Cam^i$ and Cam^j ;

elseif $((F1 = 0)$ and $(Fi = 0)$ and $(Fj = 1))$ or $(F2 = 1)$ then

$Cam^1 := Cam^i$; $Cam^2 := Cam^j$;

else $Cam^1 := Cam^j$; $Cam^2 := Cam^i$; endif;

$m := m + 1$;

end;

endif;

end Bit-Serial SUM;

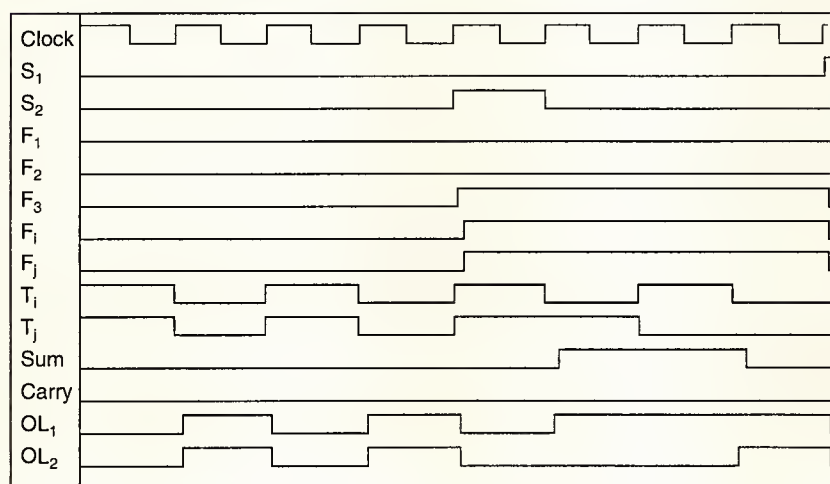


Figure 7. Simulation of the example in Figure 6.

3 in the Aggregation Functions box illustrates the AVERAGE operation, which is a combination of both Sum and Count. The AVERAGE operation proceeds as follows. To each reduced tuple (Ca, Mb, Cg) , we attach a new attribute value Cc . This attribute value is set initially to 1. The augmented tuple (Cc, Ca, Mb, Cg) is input one bit at a time to the Sum unit. The Cc value is input immediately following input of the Ca values. The first $(k+1)$ bits output from the Sum slice processor represent a Group-By value followed by a mark bit. The next p bits output from the unit represent a Ca value. Finally, the remaining output bits represent a Cc value.

Designing a Sum processor

A one-bit slice processor can process two tuples one bit per tuple at a time.

Aggregation functions

In database management applications, we often want to categorize the tuples of a relation by the values of a set of attributes and extract an aggregated characteristic of each category. We call these database management tasks aggregation functions. For instance, the SQL data language includes the following built-in aggregation functions: Sum, Count, Average, Min, Max. The attributes used for the categorization are referred to as Group By columns.

Consider the relation *professor* as *professor (faculty, department, salary)*. Each tuple of this relation gives the name of a faculty person, the department, and the academic year salary. Table A lists an instance of the relation *professor*.

In this relation a row such as <Smith, Electrical Eng., \$39,000> is referred to as a tuple. There are no duplicate tuples. Consider the following queries:

Query 1: How many faculty are in each department? The result is to be ordered by *department*.

This query requests a count of the number of faculty for each department. *Faculty* are therefore categorized according to the attribute *department*. As a result, *department* is referred to as a Group-By attribute. In SQL, the above query is formulated as follows:

Q1: Select *department*, Count (*faculty*)
From *professor*
Group By *department*
Order By *department*

The result of applying the Count aggregation function is a new relation with two attribute names. They are a Group-By attribute (*department*, in this case) and a new attribute called Count. The tuples are ordered lexicographically in ascending order according to the Order-By attribute (*department*). The resulting relation, called *Man-power*, is

<u>department</u>	<u>Count (faculty)</u>
Computer Sc.	4
Electrical Eng.	3
Mechanical Eng.	2

Query 2: What is the total salary for all faculty in each department? The result is to be ordered by *department*.

The SQL expression for this query is

Q2: Select *department*, Sum (*salary*)
From *professor*
Group By *department*
Order By *department*

Table A. Instance of the relation *professor* (*faculty, department, salary*).

<i>faculty</i>	<i>department</i>	<i>salary</i>
Smith	Electrical Eng.	\$39,000
Joe	Mechanical Eng.	\$35,000
Susan	Computer Sc.	\$36,000
Erick	Electrical Eng.	\$38,000
Paul	Electrical Eng.	\$37,000
Johannes	Computer Sc.	\$65,000
Rick	Computer Sc.	\$32,000
Gerard	Computer Sc.	\$43,000
Kenneth	Mechanical Eng.	\$40,000

The result of the Sum aggregation function is a new relation called *payroll*.

<u>department</u>	<u>Sum(salary)</u>
Computer Sc.	\$176,000
Electrical Eng.	\$114,000
Mechanical Eng.	\$75,000

Query 3: What is the average salary in each department? The result is to be ordered by *department*.

The SQL formulation of query 3 is

Q3: Select *department*, Average (*salary*)
From *professor*
Group By *department*
Order By *department*

Applying the Average operator to the relation *professor* yields the relation *median-salary*.

<u>department</u>	<u>Average (salary)</u>
Computer Sc.	\$44,000
Electrical Eng.	\$38,000
Mechanical Eng.	\$37,500

Note that the AVERAGE aggregation operation combines both the SUM and COUNT operations. Another way of representing the relation *median* is as follows:

<u>department</u>	<u>Sum (salary)</u>	<u>Count(faculty)</u>
Computer Sc.	\$176,000	4
Electrical Eng.	\$114,000	3
Mechanical Eng.	\$75,000	2

continued on next page

Aggregation functions (continued)

The Average (*salary*) for each department can be easily obtained by performing the following operation:

$$\text{Average (salary)} = \text{Sum (salary)} / \text{Count (faculty)}$$

Notice that the COUNT operation can be performed using the Sum operation by adding an extra attribute (*extra*) to the relation *professor*. For each tuple of the new relation denoted *professor**, the extra attribute value is set to 1. An instance of the new relation appears in Table B.

Query 4. The following Sum SQL statement is equivalent to Count statement Q1:

Q4: Select *department*, Sum (*extra*)
From *professor**
Group By *department*
Order By *department*

These remarks are important from a hardware implementation viewpoint. They imply that a Sum hardware unit, if designed efficiently, can also be used to process the Count and Average aggregation functions.

From these remarks, we can see that the processing of aggregation functions involves in general two columns: Group By (*Cg*) and aggregation (*Ca*). For instance, in Q2, the

Group-By column is *department*, and the aggregation column is *salary*. A tuple consisting only of these two attributes will be referred to as reduced tuple. During the process of performing Sum and Average, a new column called the *sum column* (*Cs*) is generated. Each value in *Cs* gives the sum of all *Ca* values corresponding to a distinct *Cg* value. Also, in the process of performing Count and Average, a new column referred to as *count column* *Cc* is generated. Each *Cc* value gives the number of occurrences of a distinct value in the Group-By column.

Table B. Instance of the relation *professor (*faculty*, *department*, *salary*, *extra*).**

<i>faculty</i>	<i>department</i>	<i>salary</i>	<i>extra</i>
Smith	Electrical Eng.	\$39,000	1
Joe	Mechanical Eng.	\$35,000	1
Susan	Computer Sc.	\$36,000	1
Erick	Electrical Eng.	\$38,000	1
Paul	Electrical Eng.	\$37,000	1
Johannes	Computer Sc.	\$65,000	1
Rick	Computer Sc.	\$32,000	1
Gerard	Computer Sc.	\$43,000	1
Kenneth	Mechanical Eng.	\$40,000	1

Processing a larger number of tuples in one pass requires several slice processors connected according to Batcher's odd-even network topology.³

The odd-even network topology has been used to design several special-purpose units. Batcher used it to design a sorting network, and Sood et al.⁶ used this structure to implement relational algebra operations. Abdelguerfi et al.⁴ uses the same unit for signal processing. Here, we show how a larger Sum unit can be obtained by connecting a number of slice processors using the odd-even topology.

In this type of network topology, an *n*-input Sum unit is composed of

$$\left(\frac{n(\log^2 n - \log n + 4)}{4} - 1 \right) \quad (1)$$

identical slice processors connected according to the odd-even network topology. (Throughout this article we refer to

\log_2 as \log .) The longest path in an *n*-input Sum processing unit is composed of $\lceil \log n (\log n + 1) / 2 \rceil$ levels. You will recall that the basic component of the Sum unit is a pipelined one-bit slice processor. Slices of each level take as inputs two bits (one per reduced tuple) from the preceding level, process the two bits in one clock cycle, and output two new bits to slices

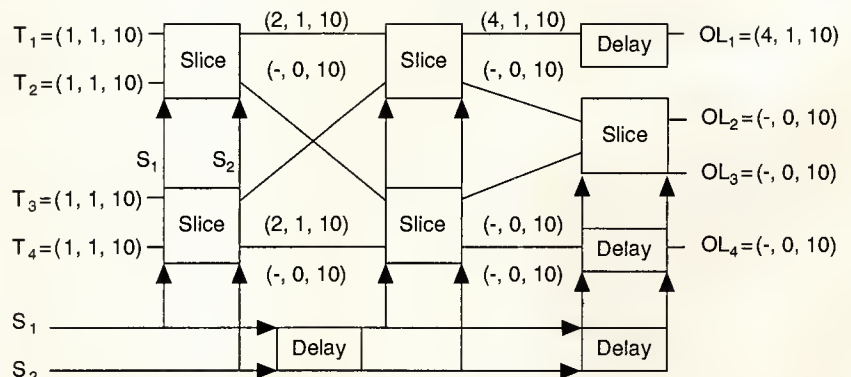


Figure 8. Example with a four-input Sum unit.

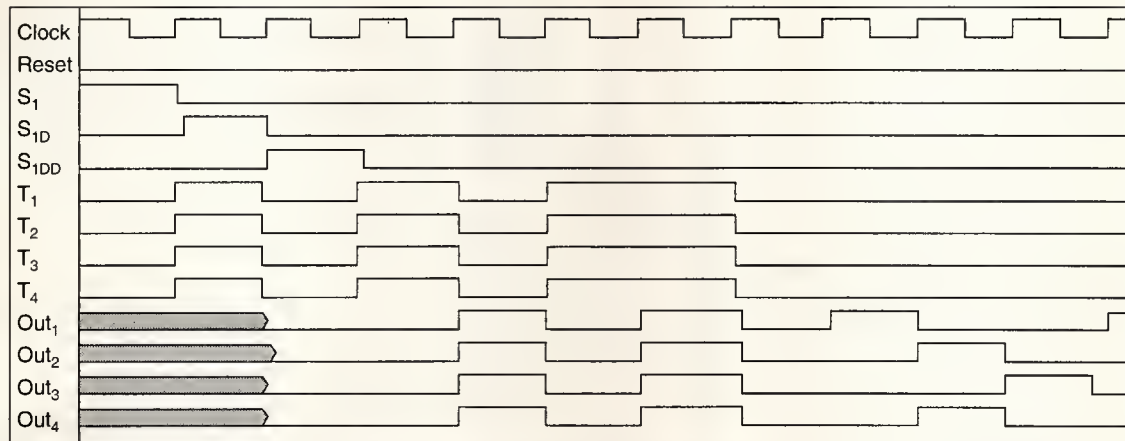


Figure 9. Simulation of the example in Figure 8.

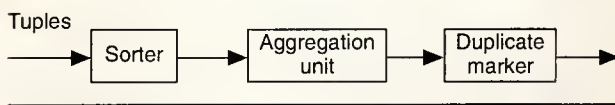


Figure 10. Processing approach of Kim et al.⁵

of the next level. We designed a four-input unit using the slice processor of Figure 4 as a basic component. Since some paths are shorter than others, a synchronization, which can be achieved through the use of delay elements, is necessary. We simulated the example in Figure 8 using this unit. The result of the simulation appears in Figure 9. Notice that the tuples are Summed, marked, and sorted concurrently. In this example, the output of the processed tuples starts three clock cycles after the input process initiation. In general, for an n -input Sum unit, the output of the processed tuples starts $\lceil \log n(\log n + 1)/2 \rceil$ clock cycles after initiation of the input process.

Comparative analysis

Kim et al.⁵ presented the design of a parallel and pipelined query processor in which tuples process in parallel, one bit at a time. Relational database aggregation functions take three steps (see Figure 10). The relation is first sorted, then fed to an aggregation unit, and the resulting relation is sent to a duplicate marker. Three different network topologies perform statistical aggregation functions.

In our design, sorting, aggregation, and duplicate marking take place concurrently. One network topology and one type of processing element (slice processor) implement these functions.

The parameters used in the comparative analysis are

- k , the number of bits in the binary representation of the

Group-By values representation;

- p , the number of bits in the binary representation of the aggregate values;
- r , the time (in seconds) to manipulate and pass one bit to the neighboring slice processor; and
- n , the number of inputs in the Sum unit.

In our implementation, the processing of the reduced tuples completely overlaps the input and output of the reduced tuples to and from the Sum unit. Since the longest path in an n -input Sum unit is $\lceil \log n(\log n + 1)/2 \rceil$ processing n tuples will take

$$A(n) = \left\lceil \frac{\log n(\log n + 1)}{2} + (k + p + 1) \right\rceil r \quad (2)$$

Notice that our approach allows for pipelined processing of different streams of tuples (different relations). (The flags of each slice processor in the first column should be initialized by applying reset signal S_1 after the input of each relation completes.) Suppose that m relations, each composed of n tuples, are to be processed by the Sum unit. The processing of these m relations in a pipelined manner will reduce the processing time from $mA(n)$ to $A(n) + (m - 1)(k + p + 1)r$. The processing is therefore reduced by

$$\left(\frac{m-1}{2} \right) \log n(\log n + 1)r \quad (3)$$

seconds.

The Sum algorithm we have described was an internal one. That is, we assumed the number of reduced tuples to be processed would be no larger than the number of inputs (n) of the Sum unit. In general, an entire relation cannot be pro-

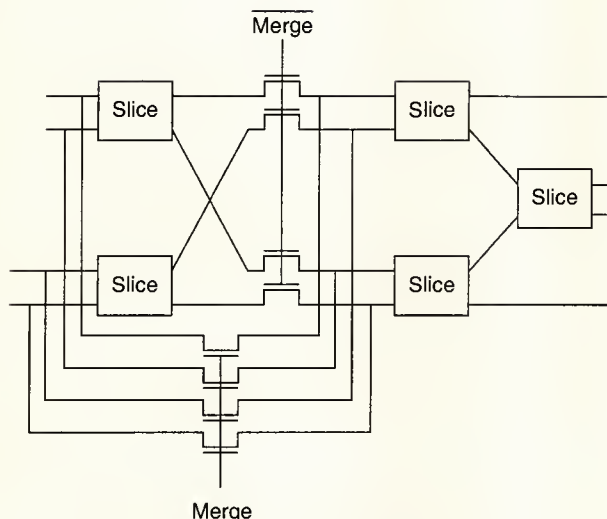


Figure 11. A Sum unit with a Merge signal.

cessed internally by an odd-even network. When the number of tuples is too large for a Sum unit to process internally, an external algorithm is the most practical solution. An external algorithm is one that allows a chip (or a set of chips) of fixed size to process an input set of any size.⁴ One approach to this problem is based on an iterative use of a Sum unit of fixed size.

We based the proposed algorithm on successively merging Summed sets of reduced tuples of increasingly larger size. The external algorithm will use a Sum unit of fixed size (n -inputs) to process in an iterative manner a set of reduced tuples whose number N is larger than n . Toward this end, we added a new control signal called Merge to the Sum unit. This signal allows an n -input Sum unit to have two operating modes. In the first mode (Merge = 0), the unit performs the Procedure Sum internal algorithm just illustrated. In the second operating mode (Merge = 1) the unit merges two Summed sets of size $n/2$ each in $\log n$ steps. Figure 11 shows a four-input Sum unit with a Merge control signal.

The external algorithm is divided into two steps. During the first step the Merge signal is reset to zero, and the Sum unit generates N/n Summed sets of size n each. (Without loss of generality, N is assumed to be a multiple of n .) When the N/n sets process in a pipelined manner, the duration of this step is

$$\frac{\log n(\log n + 1)}{2} + \frac{N}{n}(k + p + 1)r \quad (4)$$

In the second step, the histogramming unit operates as a $[(n/2) \times (n/2)]$ two-way merger by setting the Merge signal to

1. The second step requires $\log(N/n)$ phases. During the i th phase, the unit converts $1 \leq i \leq \log(N/n)$, $(1/2)^{i-1}(N/n)$ sets of $2^{i-1}n$ Summed reduced tuples to $(1/2)^i(N/n)$ sets of $2^i n$ reduced tuples each. Pipelining the $\log(N/n)$ phases yields the following duration.

$$\left(\frac{\log n(\log n + 1)}{2} + \sum_{i=1}^{\log(N/n)} \left(\left(2^{i+1} - 1 \right) / 2^i \right) (k + p + 1) \right) r \quad (5)$$

$$= \left(\frac{\log n(\log n + 1)}{2} + \left(2 \frac{N}{n} \log \frac{N}{n} - \frac{N}{n} + 1 \right) (k + p + 1) \right) r$$

When pipelining is used between the two steps, the overall duration of the external algorithm is

$$A(n, N) = \left(\frac{\log n(\log n + 1)}{2} + 2 \left(\frac{N}{n} \right) \log \left(\frac{N}{n} \right) + (k + p + 1) \right) r \quad (6)$$

In the VLSI query processor of Kim et al., tuples are processed in a tuple parallel bit-serial fashion. Unlike our approach, this processor performs aggregation functions on presorted relations. An $(n \times n)$ sorter is used to sort internally n tuples and can also be used as a $(n \times n)$ two-way merger. Designing an $(n \times n)$ sorter requires $n(3n - 1)$ bit-serial processing elements. The longest path in the sorter consists of $(2n - 1)$ stages. The time needed to sort n tuples is $(k + p + 2n - 1)r$.

This sorter requires $4n$ I/O pins, as opposed to $2n$ for our Sum unit. The duration of merging two sorted lists of n tuples each is the same as that of sorting n tuples. The time needed to sort $N \geq n$ tuples can be computed using an approach similar to the one used to compute $A(n, N)$. First we sort each N/n set of n tuples. Next, the unit is used as an $(n \times n)$ two-way merger. As Kim et al. indicate, the overall duration of the sorting is

$$\frac{N}{n} (k + p + 2n - 1) \left(1 + \log \frac{N}{n} \right) \quad (7)$$

The sorted relation is then input to the aggregation unit. An n -input aggregation unit composed of $\log n$ stages requires $n \log n$ processing elements. Processing N tuples will require about

$$\frac{N}{n} (k + p + \log n) r \quad (8)$$

The third step is the process of assigning a mark bit to each reduced tuple so that duplicates can be removed. An n -

Table 2. $B(n,N)/A(n,N)$ for $p = k = 16$ bits.

N	$N/n=2^4$	$N/n=2^8$	$N/n=2^{10}$	$N/n=2^{12}$	$N/n=2^{14}$
16	1.50	1.20	1.16	1.12	1.10
32	2.05	1.75	1.69	1.65	1.62
64	3.50	2.96	2.85	2.78	2.72


input duplicate checker composed of $(2n - 1)$ PEs organized in two stages completes this step.⁵ Processing N tuples will require about $(N/n)(k + p + 3)r$ time.

The overall duration of performing Sum is thus

$$B(n, N) = \left[\begin{aligned} &(k + p + 2n - 1) \left(1 + \log \frac{N}{n} \right) \\ &+ (k + p + \log n) + (k + p + 3) \end{aligned} \right] \frac{N}{n} r \quad (9)$$

The comparative analysis shown in Table 2 makes it clear that the odd-even approach is significantly faster than the approach in Kim et al. For example, when $N/n = 2^{10}$, the acceleration is 1.69 for $n = 32$ and 2.85 when n is increased to 64. Also, the performance gap decreases as the ratio N/n increases. We attribute this fact to the query processor sorter/merger, which requires $4n$ I/O pins, as opposed to $2n$ for the Sum unit.

OUR SPECIAL-PURPOSE, parallel bit-level, pipelined processing unit supports relational database aggregate functions. The architecture is based on the odd-even network topology, and the system is composed of one type of simple slice processor. Each slice operates on data, bit by bit, making the design and verification of the circuit easy. The data processing time is completely overlapped with the input and output of data to and from the unit. The design is independent of the tuple size, and since a bit-serial computation is used, the system requires limited interconnection.

We designed and simulated a prototype four-input unit for fabrication using discrete components. Currently, we are designing and fabricating a VLSI prototype unit and studying clock skew over long serial paths and the compatibility of the design to RAM. 

Acknowledgments

We thank all who helped us in this project. H. Mounaf provided the design and simulation. Wayne Patterson, director of the New Orleans Advanced Computation Laboratory, gave us the opportunity to use the computing facilities of his laboratory at various stages of the project. R. Loggins helped prepare the manuscript, and the referees offered comments and helpful discussions of the issues.

Parts of this article appear in the *Proceedings of the Sixth International Workshop on Database Machines* dated June 1989 and the *International Conference on Parallel Processings* dated August 1990. A grant from the University of New Orleans Research Council and the NSF/Louisiana Stimulus for Excellence in Research, EPSCoR Program under Grant NSF/LaSER(1990)-RFAP-14 partly supported our work.

References

1. E.F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Comm. ACM*, Vol. 13, No. 6, pp. 377-387.
2. M.S. Giovanni, "Fragmentation: A Technique for Efficient Query Processing," *ACM Trans. Database Systems*, Vol. 11, No. 2, 1986, pp. 113-123.
3. K.E. Batcher, "Sorting Networks and their Applications," *Proc. Spring Joint Comp. Conf.*, Vol. 32, Apr., 1968, pp. 307-314.
4. M. Abdelguerfi, S. Khalaf, and A.K. Sood, "Bit-Serial Parallel Processing Unit for the Histogramming Operation," *IEEE Trans. Circuits and Systems*, Vol. 37, No. 7, July 1990, pp. 948-954.
5. W. Kim, D. Galski, and D.J. Kuck, "A Parallel Pipelined Relational Query Processor," *ACM Trans. Database Systems*, Vol. 9, No. 2, June 1984, pp. 214-242.
6. A.K. Sood, M. Abdelguerfi, and W. Shu, "Hardware Implementation of Relational Algebra Operations," in *Database Machines: Modern Trends and Applications*, NATO ASI, Series F, Springer-Verlag, Berlin, 1986, pp. 321-380.

The authors' biographies, pictures, and addresses appear in the Guest Editors' Introduction on p. 7.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 165

Medium 166

High 167



A Parallel, Scalable, Microprocessor-Based Database Computer for Performance Gains and Capacity Growth

The multiback-end database supercomputer, or MDBS, consists of the architecture and performance of an experimental database computer and a number of database processors and their corresponding database stores. The author relates two studies: one on the design goals and architectural considerations of the microprocessor-based MDBS and the other on the performance expectations and benchmark results in various loads and configurations.

David K. Hsiao

Naval Postgraduate School

The MDBS architecture is unique in that each database processor is microprocessor-based and has its own database store. These stores consist of two different types of disk drives: smaller, Winchester-type drives for paging and metadata (keys and key-related data); and a larger, standard-type drive for base data (database data). The multiple database processor-store pairs, known as database back ends, interconnect through a local area network, with reliable point-to-point communications (one processor to another) and unreliable broadcast communications (one processor to many others).

The interconnected database back ends interface with another computer either directly via the LAN or indirectly via the database controller. With a direct interface, the database controller software runs in another computer known as the front end or server. With an indirect interface, the database controller is a microprocessor-based computer that uses either a tape station or built-in cassette tapes.

We use MDBS as a research vehicle in the Laboratory for Database Systems Research at the Naval Postgraduate School for the study of the design and performance of parallel and scalable database back ends. At present, MDBS can be configured or scaled into a one-back-end, two-back-end, and up to an eight-back-end database computer for parallel operations and performance analyses.

The performance gain of MDBS is unique in that the response-time reduction of a transaction is in direct proportion to the number of back ends configured. For example, if we double the number of back ends in a parallel configuration, we reduce the response time of the same transaction in the same database by almost half. Although the database remains the same in both configurations in this example, it must be redistributed to induce parallel access to any new, as well as existing database stores. We use response-time reductions to measure the performance gains of the database computer compared to the degree of its parallelism or the number of parallel back ends used.

The growth capacity of MDBS is unique in that the response-time invariance of a transaction may be nearly upheld if the increasing degree of back-end parallelism is in direct proportion to the growing capacity of the database. In other words, if we want to have nearly the same response time for a transaction as its database doubles, we simply double the number of parallel back ends. Again, we must recluster and redistribute the grown database on the new as well as existing back ends to induce parallel accesses to all the database stores. We use response time invariance to provide nearly constant response times for the same transactions, despite database growth. We simply add parallel back ends proportionally.

Finally, in this article, we point out some of

the difficulties and examine some unfinished studies that may have some impact on the architecture of parallel database back ends in general, and MDBS in particular. We have learned these lessons in the course of experimenting with this new architecture. We hope our architectural design and performance methodology will serve as a basis for the design and construction of future parallel, scalable, and microprocessor-based database computers.

The MDBS architecture

Figure 1 depicts the MDBS architecture. The major building block for the high degree of parallelism is the database back end. To achieve a high degree of parallelism, we simply add identical back-end hardware to the LAN, replicate the back-end software and metadata on the new hardware, recluster as necessary, and redistribute the base data onto the new and existing database stores. Let us elaborate on the major building block first. We can then discuss the rest of the architectural elements.

Database back ends. Eight database back ends are available in our laboratory. Although we are not specific in Figure 1, all eight back ends are identical. We therefore focus on the design of a single back end in the following sections.

Hardware elements. Each database back end is a combination of a database processor and a database store.

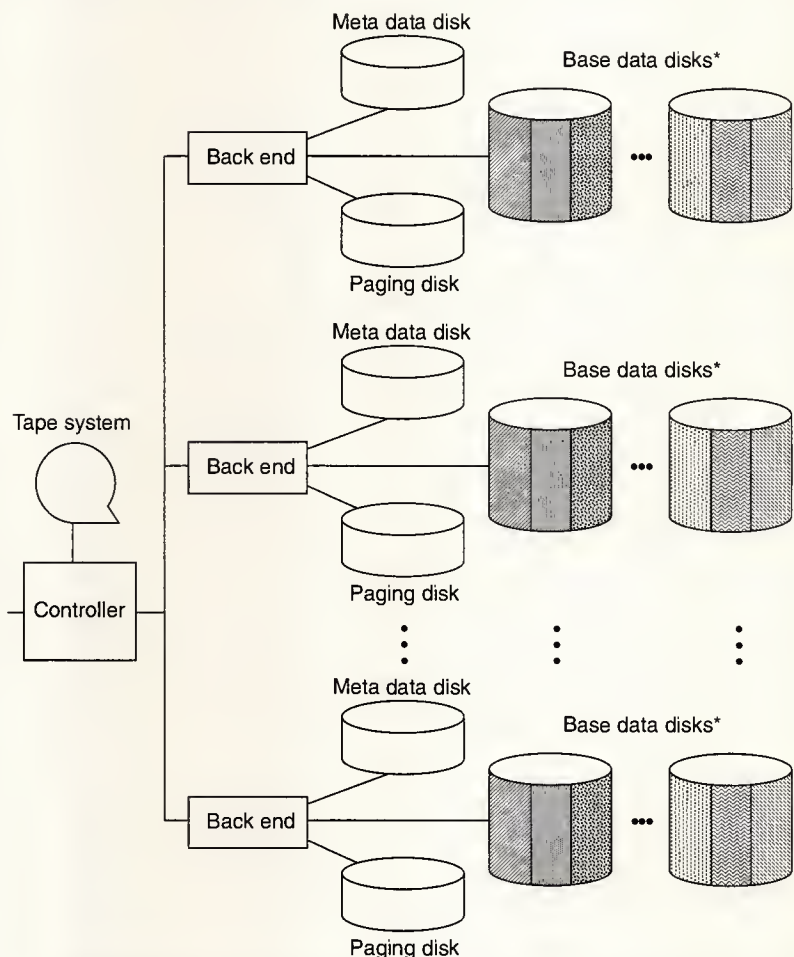
- **CPU and internal data bus.** Each database processor consists of a Motorola 68020 microprocessor-based CPU with internal 32-bit-wide data and control buses. At 16.67 MHz, the 68020 is adequate for database operations, since the operations are mostly I/O-intensive rather than computationally intensive. Further, for any data transfer between the real memory and external buses, we use the VMEbus. The VME bandwidth is 15.5 Mbytes/s. On the other hand, the real memory cycle of the database processor is 240 ns per 32-bit word. We use the following calculation:

$$\begin{aligned} 4/240 \text{ bytes/ns} &= 4 \times 10^9/240 \text{ bytes/s} \\ &= 16.67 \times 10^5 \text{ bytes/s} = 16.67 \text{ Mbytes/s.} \end{aligned}$$

Thus, since 15.5 Mbytes/s is below its 16.67-Mbytes/s capability, the maximum amount of data that can be concurrently transferred in (and out) of real memory

for processing over four buses is nearly covered by the bandwidth of the VMEbus. The buses consist of three for I/O—paging, metadata, and base data—and one communications bus for the LAN. There are still some memory cycles left (at the rate of 11.17 Mbytes/s) for the CPU control bus to access the real memory.

- **Backplane and communication bus.** All four external buses do not need to access the VMEbus simultaneously. On those rare occasions when competition among memory cycles for real-memory access is keen, the CPU control bus always gives the right-of-way to I/Os and communications. Further, the communications bus, known as the backplane of the back end, is managed by another microprocessor with its own buffer memory, and also gives the right-of-way to I/O buses.



* Tracks with the same shading consist of records belonging to the same cluster.

Figure 1. Architecture of MDBS.

We discuss the backplane hardware later.

- **External I/O buses and the internal VMEbus.** As discussed, the I/O buses can monopolize the entire bandwidth of the VMEbus. The microprocessor-based CPU and real memory can thus sustain disk drives whose maximum transfer rate is about 16 Mbytes/s.
- **Disk drive types.** Our system contains three disk drives; one is for paging, another for metadata, and the third for base data. Because the 68020 microprocessor supports address translation, a larger virtual memory (from 2 Mbytes to 32 Mbytes) may be supported by a smaller, 4-Mbyte real memory. There is still, however, need for a paging disk for virtual memory pages. MDBS uses a 96-Mbyte Winchester-type disk drive for paging, because most of the operating and database system modules are real memory-resident and their pages are locked into the real memory. We use another 96-Mbyte disk drive for the metadata. You will recall that database metadata are keys and key-related information. Queries use keys to access clustered records directly. The number of keys used relates to the average size of the database clusters, which in turn relates to the number of back ends in the configuration. Finally, a large, standard, 500-Mbyte moving-head disk stores the base data. Here, the key-to-record ratio of metadata storage to base data storage is 96:500, nearly 1:5 per back end.
- **Disk transfer rates.** The transfer rate of each disk drive is under 2 Mbytes/s. Thus, with three drives transferring simultaneously in or out of real memory, the maximum rate is under 6 Mbytes/s. On the other hand, real memory can sustain a rate of some 16 Mbytes/s, so there is enough capacity for the addition of, for example, two more sets of metadata and base data disks.
- **Database store.** For each back end there may be 288 Mbytes of metadata and 1.5 Gbytes of base data for its database store. For the eight-back-end database computer in our laboratory, there is a potential for some 12 Gbytes of database data. On the other hand, the metadata are replicated. Thus, the real capacity of metadata remains at 288 Mbytes. The key-to-record ratio drops from 1:5 in the one-back-end configuration to nearly 1:40 in the eight-back-end configuration. To maintain the capacity of metadata relative to the capacity of base data, we may add more 96-Mbyte metadata disks and fewer 500-Mbyte base data disks. In other words, the database store is scalable for direct access, depending on the capacity of the metadata and base data. The database back end is also scalable. In our laboratory, it spans from one to eight.

Software processes. Since a database is considered as residing in a back end, we use the Directory Management process to manage metadata. We also use a process for managing

base data, the Record Processing process. In each back end, transactions execute concurrently. Metadata and base data processing of a transaction take place serially, and the metadata processing of a transaction may overlap with the base data processing of another transaction. And, since metadata processing as well as base data processing for different transactions may take place concurrently, we need a Concurrency Control process.

A back end must be able to receive transactions for processing and return responses. Thus, each back end has a pair of processes for communications: the Get process, for getting transactions or messages from the LAN, and the Put process, for placing responses or messages on the LAN. Directory Management, Record Processing, Concurrency Control, Get, and Put are the only processes of a back end. These processes are the same in every back end in a multiback-end configuration. Further, these five database management and communication processes use a Unix BSD 4.3 operating system with TCP/IP protocols.

A microprocessor-based LAN. Through its Get and Put processes, the microprocessor-based LAN is the only communications link between the controller and back ends.

LAN hardware elements. These elements are an Ethernet cable, the backplanes, and their transceivers. Each back end contains a backplane that consists of an Intel 80186 microprocessor and a small amount of real memory. The 128-Kbyte real memory is used mainly for buffering incoming or outgoing messages. It is also dual-ported to another microprocessor (Intel 82586) that mainly executes low-level protocols, which in turn support high-level TCP/IP protocols. These protocols in turn support even higher protocols, such as UDP (Ethernet's user-defined protocol).

The Ethernet cable connects all the backplanes by way of transceivers. It thus enables all 82586 microprocessors to work in concert to execute a collision-resolution algorithm whenever there is a collision or conflict of messages. One of the messages is then selected for routing, while all other messages are deferred momentarily and considered for routing again. If, in the next routing, there is another collision, the algorithm executes again. Eventually, all messages will be routed to their designations so no messages are lost.

This algorithm causes restricted point-to-point (one backplane to another backplane) routing. In other words, if, for example, three point-to-point routings collide (say, from Backplane 1 to Backplane 4, from Backplane 3 to Backplane 8, and from Backplane 7 to Backplane 2), the three pairs route one after another, although we cannot predict the routing sequence. We note that, in this example, all backplanes involved in the routing are disjointed, and no two pairs overlap on an identical backplane. The Ethernet cable has a transmission rate of 10 Mbps.

On the other hand, the 82586-based Ethernet does not provide reliable broadcasting (one backplane to many

backplanes routing). For example, if Backplane 2 wants to broadcast a message to all other backplanes, namely, Backplanes 1 and 3 through 8, the routing takes place. However, the message may not get to one or more of the other backplanes, let's say, Backplane 7. This kind of unreliable routing is also unpredictable. Lastly, the 82586-based Ethernet does not provide reliable multibroadcasting either.

LAN software protocols. Since the LAN's own TCP/IP software does not support reliable broadcasting and multibroadcasting, we built another layer of protocols, UDPs, for these operations.

At each back end, a logical socket for each and every other back end is defined. Whenever a message reaches a particular back end, that back end must acknowledge receipt. The acknowledgment is deposited in its corresponding socket at the sender's back end. This send-receive-acknowledge procedure via a socket is not necessary if the message routing is point-to-point (one back end to another). In other words, the acknowledgment is always there and can be ignored.

***Our investment in the software
is minimal, because it involves
only a few memory locations
for sockets.***

In broadcasting a message to other back ends, the UDP software examines the acknowledgments deposited at the sender's back end. If there is no acknowledgment at a specific socket, the program resends the message to the corresponding back end's socket by way of point-to-point routing. Obviously, the receiver's back end cannot begin its database operation simultaneously and in parallel with other back ends that had an earlier start on the same message. MDBS's back ends are not tightly coupled, do not share primary and virtual memories, and have their own database stores. Thus, all other back ends can proceed individually and in parallel without any interference or delay from one or more identical back ends.

The UDP software helps make unreliable broadcasting reliable. Our investment in the software is minimal, because it involves only a few memory locations for sockets. The use of existing and reliable point-to-point protocols for resending messages is overhead, but it does not interfere with other back ends that require no resending. So, with reliable broadcasting, we now also have reliable multibroadcasting—a communication feature necessary to our back ends. The back ends

and controller access the UDP software with Get and Put.

Database controller. As stated previously, the controller software may run in a general-purpose front-end computer that interfaces with the database back ends via the LAN. However, because it does not interfere with other program activities and programs in the front-end computer, for research and experimental purposes, a dedicated database computer like MDBS is preferred.

Controller hardware. Like a back end, the controller hardware is based on a 68020 microprocessor with address translation capability, so we still need a paging disk. Unlike a back end, the controller accesses neither metadata nor base data. On the other hand, the controller is responsible for the backup and recovery of the back end and LAN software. Thus, we place a tape station at the controller. Backup and recovery tapes for on-line and real-time backups and recoveries can be readily made. We also plan to use tapes to load a massive new database into back-end database stores. In addition, we use tapes to load benchmarks for performance analyses.

Controller software. Because the controller computer is the sole interface with the "outside" world, it needs two processes: one for receiving database transactions from the front end and another for returning database results to the front end. These processes are known as TP (Request or Transaction Processing) and PP (Postprocessing).

A user request may be a new database transaction requiring compilation, a canned transaction or macro in a transaction library, or an on-line query requiring immediate response. TP must be able to uniquely identify each request, preprocess it, and broadcast the request to all back ends. Each back end, on the other hand, places a broadcast request in its request (or transaction) queue. The preprocessing of each transaction amounts to a translation of the transaction into one or more back-end primary database operations. We discuss the five primary database operations, RETRIEVE, DELETE, INSERT, UPDATE, and RETRIEVE COMMON, in a later section on databases and their operations.

The PP process, on the other hand, is the postprocessing of database records. For example, if a user desires the sum of all the salaries from the salary database, each back end returns a subsum of all the salaries from its portion of the salary database. Then, PP adds all the subsums to produce the overall sum that is then routed to the user. In this example, PP performs the aggregate function, Sum. In general, PP performs a large number of aggregate functions. By interfacing with TP, PP can uniquely identify the correct user to return the result or error message.

TP and PP interface with the outside world. We use an Insert Information Generator, or IIG, process designed solely to assist a back end during an INSERT operation in the "inside" world of the database computer. Of the five primary back-end database operations, the INSERT operation is the

Upon insertion of a record, the particular cluster in which the record belongs must be identified. However, the cluster is evenly distributed over a number of back-end disk stores. The database store that provides the latest available storage space for any record insertion is identified in a space utilization table maintained by the IIG. The space utilization table keeps the following information up to date. For each cluster of records, it identifies the back end whose database store contains the first trackful of the cluster, and it identifies the back end whose database store contains the last trackful of records. It also identifies the back end whose database store can provide the first available track for inserting the new trackful of clustered records. The maintenance and allocation by the IIG are based on a round-robin algorithm explained later. In any case, with the help of the space utilization table, the IIG instructs a specific back end to insert records into its database store. The actual insertion is made through the processes of the specific back end. Further, since metadata appear in every back end, the specific back end must also broadcast its metadata updates to all other back ends for replication.

Architectural elements and system processes. In Figure 2, we illustrate the relationship between the elements and the system processes. Since all the architectural elements and system processes in a back end are identical to the ones in another back end, we depict those in only one back end. We observe that, for communications, the Put process of one computer sends messages to one or all Get processes in the other computers. Thus, Put can facilitate either one-to-one or broadcasting in its communication with the other computers in the system.

Figure 1 illustrates the architecture of the system, divided into two main sections: the Front-end computer and the Back end.

Front-end computer:

- Controller:** Contains the TP (Transaction Processing) and PP (Postprocessing) modules.
- TP (Transaction Processing):** Receives data from the 'From' source and sends data to the PP module.
- PP (Postprocessing):** Sends data to the 'To' destination.
- Put and Get modules:** These modules are connected to the TP and PP modules. The TP module sends data to the Put module, and the Get module sends data to the PP module.
- IIG (Insert information generator):** This module is connected to both the Put and Get modules.

Back end:

- Get module:** Receives data from the Put module in the Front-end computer via the LAN.
- CC (Concurrency Control):** Receives data from the Get module and sends data to the DM (Directory Management) and RP (Record Processing) modules.
- DM (Directory Management):** Receives data from the CC module and sends data to the RP module.
- RP (Record Processing):** Receives data from the DM module and sends data to the Put module.
- Put module:** Sends data to the 'To other back ends' destination.
- Meta data store:** Connected to the DM module.
- Base data store:** Connected to the RP module.

Legend:

- CC Concurrency Control
- DM Directory Management
- IIG Insert information generator
- PP Postprocessing
- RP Record Processing
- TP Transaction Processing

Data organization and repertoire of operations

- models metadata and base data separately;
- introduces an equivalence relation, which partitions the database into mutually exclusive sets of base data called clusters; and
- allows clustered records to be distributed in back ends, which induces parallel access to database stores.

The rationale for implementing ABDL instead of a conventional data language such as SQL, derives from the following intrinsic properties of ABDL. ABDL

- supports a parallel search algorithm based on predicates, and
- is semantically rich and complete so that transactions written in conventional data languages like SQL may be translated into ABDL for execution in MDBS.

These and other properties of ABDM and ABDL are discussed later.

Base data and metadata. Every piece of data in the database is characterized in ABDM as an attribute-value pair. Thus, when we refer to a piece of data, say, USA, in an example database, we not only refer to its value, USA, we also know its attribute, Country. An attribute-value pair is formally denoted as an ordered pair enclosed in angular brackets, <Country, USA>. Attribute-value pairs are the building blocks of the database.

Base data. A record is a set of attribute-value pairs such that no two attribute-value pairs of a record have the same attribute, and at least one attribute of a record is typed. The first rule ensures that any attribute of the record is single-valued. The second rule ensures that at least one key identifies the record. A record is formally denoted as a set of attribute-value pairs enclosed in parentheses: (<File, aircraft>, <Classification, top-secret>, <Plane-Type, fighter>, <Radius, 799 nautical miles>, <Country, USA>, <Fuel-capacity, 600 gallons>, <Pilot-Grade, test-pilot>). All the records of the database comprise its base data. In reality, there are thousands, if not millions, of base data or records, in the database.

Metadata. There are three attribute types. The Type A attribute is one of disjointed value ranges. Thus, a Type A attribute partitions its values into value ranges. For example, the attribute Radius in Table 1 is a Type A. A Type B attribute is one of distinct values. For example, the attribute Country in Table 1 is a Type B, since it has two distinct values, USA and USSR (shown in Table 2). A Type C attribute is one of those distinct values being entered by the user in real time. Thus, Type Cs are like Type Bs in assuming distinct values. However, Type B attributes are created at the time of database creation or at generation time. Attributes and their types collect in the Attribute table (Table 1). For a real-world database, an attribute table consists of tens, if not hundreds, of attributes. We illustrate only five in Table 1.

We call each Type A attribute and its value ranges Type A descriptors. Similarly, we call Type B and Type C attributes and their distinct values Type B and Type C descriptors. If we relate the descriptors here to conventional keys, we have three different kinds of keys. Whereas both Type A and Type B descriptors tend to remain the same, because of their creation at database generation time, the Type C descriptors

may increase rapidly if a user enters records consisting of many new values having the Type C attribute. All the descriptors collect in a table known as a descriptor-to-descriptor-identifier table, or for short, the Descriptor table (Table 2). For a real-world database, a descriptor table consists of hundreds, if not thousands, of descriptors. In Table 2, we illustrate only 16. Specifically, Radius results in six Type A descriptors; Plane-Type, in three Type C descriptors; Country, in two Type B descriptors; File, in one Type B descriptor; and Classification, in four Type C descriptors.

Let D_i be the set of D_{ij} for all j . Then the product $D_1 \times D_2 \times \dots \times D_n$ is an equivalence relation whose members partition the database data into mutually exclusive sets of records, or attribute-value sets. These record sets we term clusters.

Referring to our sample in the Attribute and Descriptor tables, we note there are five attributes. Thus, $n = 5$. For D_1 , or Radius, we see six descriptors. Thus, for $i = 1, j_i = j_1 = 6$. Similarly, for $i = 2, j_2 = 3$ for $D_2 = \text{Plane-Type}$; for $i = 3, j_3 = 2$ for $D_3 = \text{Country}$; for $i = 4, j_4 = 1$ for $D_4 = \text{File}$; and for $i = 5$,

Table 1. Attributes.

Attribute	Attribute type	DDIT entry
Radius	A	D11
Plane-type	C	D21
Country	B	D31
File	B	D41
Classification	C	D51

Table 2. Descriptors.

ID	Descriptor
D11	$0 \leq \text{Radius} \leq 400$
D12	$401 \leq \text{Radius} \leq 600$
D13	$601 \leq \text{Radius} \leq 800$
D14	$801 \leq \text{Radius} \leq 1,000$
D15	$1,001 \leq \text{Radius} \leq 1,200$
D16	$1,201 \leq \text{Radius} \leq 2,000$
D21	Plane-Type = fighter
D22	Plane-Type = bomber
D23	Plane-Type = reconnaissance
D31	Country = US
D32	Country = USSR
D41	File = aircraft
D51	Classification = top-secret
D52	Classification = secret
D53	Classification = confidential
D54	Classification = unclassified

$j_5 = 4$ for $D5 = \text{Classification}$. The cardinality of the product $D1 \times D2 \times D3 \times D4 \times D5$ is 144, since $j_1 \times j_2 \times j_3 \times j_4 \times j_5 = 6 \times 3 \times 2 \times 1 \times 4 = 144$.

Potentially, this database may have up to 144 clusters, each of which is uniquely identified by a set of descriptor identifiers, with corresponding descriptors characterizing the cluster records. In reality, many clusters are empty, since there are no records in the database being interpreted by the descriptor identifier sets determining the clusters. MDBS does not track empty clusters. Nor does it record their descriptor identifier sets in the Descriptor Table. The three kinds of identifiers are the

- 1) system-generated *record*, which identifies the records of a cluster characterized by a descriptor identifier set;
- 2) *cluster*, which identifies the cluster containing the records; and
- 3) *descriptor*, which identifies the descriptor-identifier set that determines the cluster.

These identifiers reside in the table known as the database cluster definition table, or, for short, the Cluster table. In our sample database, the Cluster table (Table 3) identifies 12 clusters of 33 records. Each cluster is defined by a unique set of descriptors. For example, the first cluster C1 is defined by five descriptors whose identifiers are D13, D21, D31, D41, and D51. Three records in that cluster have their own record identifiers, R1, R2, and R13. Despite their differences, these records all consist of top-secret aircraft data (defined by D51 and D41) about US fighters (identified by D31 and D21) hav-

ing a combat range of 601 to 800 nautical miles (D13). In practice, a cluster table has tens, if not hundreds, of entries.

The three database tables, Attribute, Descriptor, and Cluster, contain the database metadata. Since the product $D1 \times D2 \times \dots \times Dn$ induces an equivalence relation, no record identified in the Cluster table can belong to two different clusters. Furthermore, records in each cluster are unique. The equivalence relation is the foundation for our database parallel access and storage strategies.

Distribution. The distribution of metadata and base data to their separate database stores takes place differently, although both types attempt to access their stores in parallel.

Metadata. Since metadata is typically one or two orders-of-magnitude smaller in size than base data, we decided to replicate the metadata onto each of the back-end database stores. We also use the smaller, dedicated disk drives for storage of replications. Subsequently, all the back ends can access their own metadata stores and the same sets of Attribute, Descriptor, and Cluster tables, in parallel.

Furthermore, parallel metadata accesses may be overlapped with the parallel base data accesses for any other transaction, since base data stores use separate and larger disk drives. Therefore, MDBS achieves concurrent and parallel executions of transactions.

Base data. Base data make up the bulk of a database. Therefore, they are not replicated for storage. Further, they are stored on each back end's own high-capacity disk drives in a prescribed fashion. The method is as follows:

- 1) from the Cluster table, the controller picks up a cluster identifier and its associated records;
- 2) the controller blocks a variable-size cluster into fixed-size trackfuls of records;
- 3) the controller determines the identifiers of the back ends, each of which can provide a track of available storage for the cluster identified (recall that the controller IIG has a storage utilization map to keep track of such information);
- 4) the controller sends in parallel all the trackfuls of records to the back ends identified;
- 5) each identified back end places its block of one or more trackfuls of clustered records into its base data store and enters identifiers of records stored onto the replicated Cluster table entry corresponding to the cluster on the metadata store;
- 6) the controller then updates its space utilization map with respect to this cluster; and
- 7) the entire procedure is repeated for all subsequent clusters.

This turn-taking and one-track-per-back-end database distribution (or redistribution) algorithm has a desirable effect in that records of a cluster are evenly distributed (or redistrib-

Table 3. Clusters.

ID	Descriptor ID set	Record ID
C1	{D13, D21, D31, D41, D51}	R1, R2, R13
C2	{D12, D21, D31, D41, D51}	R8, R9
C3	{D14, D21, D32, D41, D51}	R20
C4	{D13, D21, D32, D41, D51}	R17
C5	{D12, D21, D32, D41, D51}	R21, R24, R27, R30, R31
C6	{D13, D21, D31, D41, D52}	R3, R4, R14
C7	{D12, D21, D31, D41, D52}	R10, R11
C8	{D13, D21, D32, D41, D52}	R18, R19
C9	{D12, D21, D32, D41, D52}	R22, R23, R25, R26, R28, R29, R32, R33
C10	{D13, D21, D31, D41, D53}	R5, R6, R15
C11	{D12, D21, D31, D41, D53}	R12
C12	{D13, D21, D31, D41, D54}	R7, R16

uted) over a set of separate, parallel database stores. Subsequent accesses to the records of a cluster become parallel. In other words, we have induced the record-parallel and cluster-serial, or RPCS, operation for our database access operations.

The choice of the cluster size is important in a parallel access operation. Let the number of record tracks in a cluster be m and the number of back ends n . Then, the maximum number of record tracks for a cluster should also be n , for example, $m < n$. To fine tune the cluster size, we control the number of descriptors used in the identification of the clusters. In general, the more descriptors used, the smaller the clusters become. If MDBS has many n and few m , say, $3m = n$, its back ends can access several (three) clusters in parallel. Thus, we have induced record-parallel and cluster-parallel, or RPCP, operation for our database access operations. Figure 1 depicts both RPCS and RPCP operations for a sample distribution of clusters of the database.

Database operations. A database transaction is composed of one or more of the five primary database operations, RETRIEVE, DELETE, INSERT, UPDATE, and RETRIEVE COMMON (see Figure 3). Except INSERT, which inserts one new record at a time into an existing database, these operations are set-oriented. More specifically, RETRIEVE, DELETE, and UPDATE operate on one set of records at a time, whereas RETRIEVE COMMON operates on two sets of records at a time.

The query—a Boolean expression of predicates. The input for all set-oriented primary database operations is a query. The output of an operation is a set of records that has satisfied the query and has been operated on. The query allows us to specify the properties of a record set without having to provide the record addresses of the set. These properties are specified in terms of a Boolean expression of predicates. A predicate is a 3-tuple, consisting of an attribute, a relational

operator, and a value. The set of relational operators include $=$, $<$, $>$, \leq , and \geq . Boolean operators include \wedge (And), and \vee (Or). The following is a query consisting of the conjunction of two predicates (one less-than-or-equal-to predicate and one equality predicate) and one disjunction of three equality predicates.

(((Radius \leq 600) \wedge (File = aircraft) \wedge ((Classification = secret) \vee (Classification = confidential) \vee (Classification = unclassified))))

Transaction processing in the controller transforms all queries into their equivalent disjunctive normal forms for set-oriented primary operations.

Parallel predicate search algorithm. In Figures 4 and 5 on the next two pages, we depict our parallel search-for-records algorithm on the basis of a given query. To simplify the illustration, we use a simple query as a conjunction of two predicates instead of the disjunctive normal form of many predicates. Given a query, the back ends can determine the descriptor sets that satisfy the predicates of the query, since descriptors and predicates are similar in form. They do differ in where they appear. Descriptors are held in metadata or Attribute and Descriptor tables, and predicates in user queries. Once determined, we use these descriptor sets to compute the descriptor-identifier sets, which in turn allow us to determine the clusters characterized by them in the Cluster table. Thus, at the time the parallel search algorithm is being used, two input parameters have been given to the algorithm: the query and the cluster identifiers from which records will be searched for checking against the query.

Let us observe the algorithm with the following sample query in Figure 4:

- 1) The controller broadcasts the query to all the back ends for execution.
- 2) Each back end uses the query to determine those clusters that may have records satisfying the query. Each back end's Descriptor and Cluster tables aid this determination. The cluster number and identifiers determined for the query are now known by all back ends.
- 3) All back ends access one or more clusters of records. Since clustered records are evenly distributed over the back ends' disks and the accesses are either RPCS or RPCP operations, parallel database streams flow to the back ends from their respective database stores.
- 4) Each back end does the following: It compares the attribute of the first predicate of the query with the attribute of the first attribute-value pair of the forthcoming record. If the attribute matches, the back end checks whether its attribute value satisfies the value of the predicate dictated by the relational operator of the predicate. If satisfied, it repeats the comparison and checks for the

RETRIEVE (query) [target-list] [by-clause]

DELETE (query)

UPDATE (query) (modifier)

RETRIEVE (query-one) [target-list-one]

COMMON (attribute-one, attribute-two)

RETRIEVE (query-two) [target-list-two]

INSERT (<attribute-first, value-first>, <attribute-next, value-next>, . . . , <attribute-last, value-last>, {any arbitrary string})

Figure 3. Five primary operations.

next predicate and attribute-value pair until there are no more predicates in the query. In this case, the record satisfies the query and is routed as output. If any one of the following cases takes place, the record is not output, and the search begins with the next record in the stream. These cases are: the first comparison for identical attributes fails; the next comparison fails the relational operator test; or the record under consideration runs out of attribute-value pairs before the query runs out of predicates for comparison.

The parallel search algorithm makes one sweep of all the database streams coming from the database stores without the need to recall those records having been swept. Our encoding technique makes this possible. First, we use attribute identifiers, as illustrated in the Attribute table, in lieu of attribute names in both attribute-value pairs and predicates. We then order the attribute-value pairs monotonically by their attribute identifiers and query predicates.

The effect of this algorithm is the creation of a single-query/multiple-database-streams, or SQMD, operation. If, at a back end's database store, there is no clustered data for a query, the back end chooses the next transaction in its transaction queue to execute. The query of the new transaction is also new. Thus, we achieve a multiple-query/multiple-database-streams, or MQMD, operation, too.

MQMD operations are the best we can achieve in parallelism for a parallel database computer. However, it should be observed that the letter "Q" in SQMD and MQMD replaces the letter "I" (instruction) in SIMD and MIMD, which are the two best parallel operations in a supercomputer. Note also that the letter "D" in SQMD and MQMD, differs from the letter "D" in SIMD and MIMD. Our "D" stands for database streams coming from their respective base-data disks, while the classical "D" stands for data items coming from main memories. To sustain parallel database streams, we cluster data and distribute the clustered data evenly across multiple disk stores. Our parallel algorithms achieve

(a)

Employees	1	2	3	4	5	6
	Employee number	Employee name	Department name	Salary	Vacation earned	Vacation used
1.	10	Smith	Sales	10000	10	0
2.	15	Jones	Sales	30000	5	6
3.	21	Brown	Purchasing	22000	12	0
4.	35	Smith	Marketing	30000	9	2
5.	42	Smith	Purchasing	20000	8	5
6.	50	White	Purchasing	25000	0	0
7.	62	Gray	Marketing	10000	4	2
8.	71	Hall	Sales	15000	10	3
9.	75	Green	Sales	10000	8	9

(b)

Field

Figure 4. Parallel search algorithm by predicates: record template (a); comparison of bit serial and database stream parallel and single sweep (b).

SQMD and MQMD, as well as RPCS and RPCP operations over the database.

RETRIEVE COMMON. As mentioned earlier, RETRIEVE, DELETE, and UPDATE are three database primary operations, each of which operates on one set of records at a time. A user query specifies the set of records to be operated on. On the other hand, RETRIEVE COMMON operates on two sets of records, each of which is specified by a query. Thus, a RETRIEVE COMMON operation involves two queries. Es-

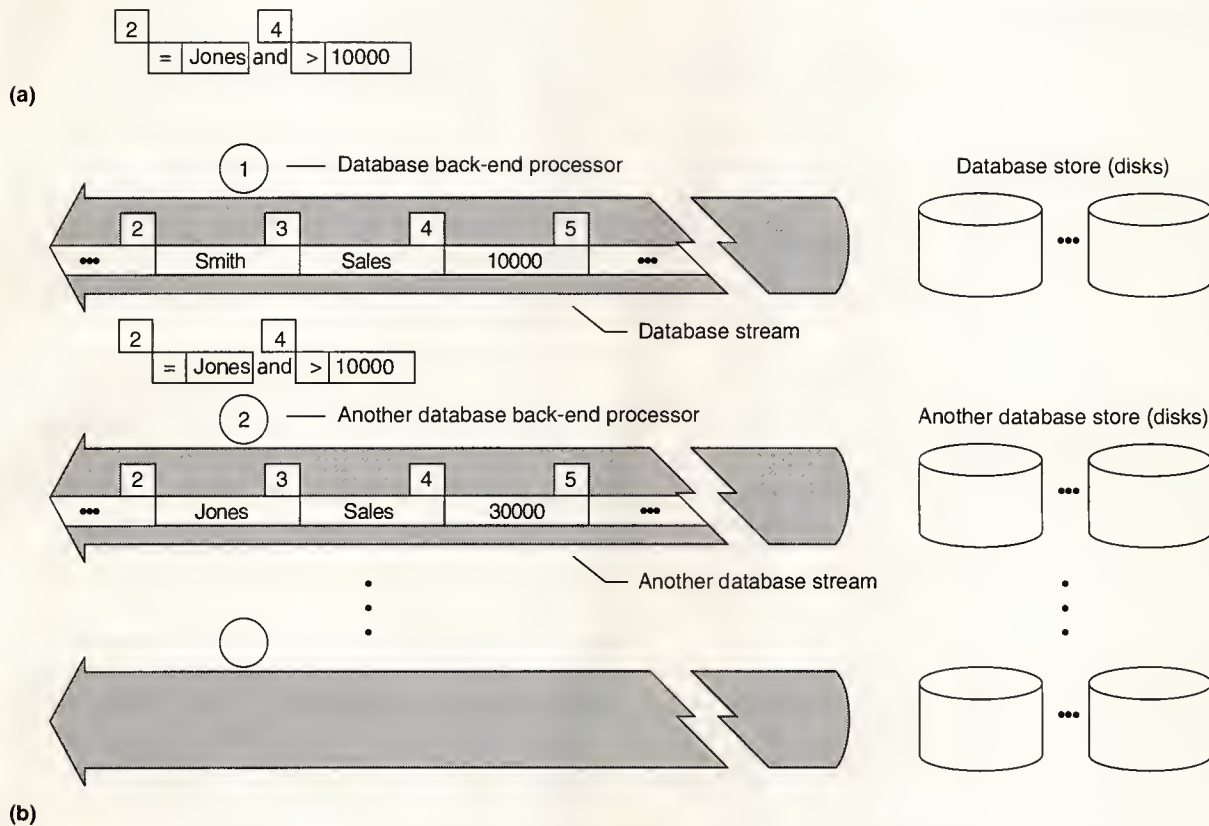


Figure 5. Conjunction of an equality predicate and a greater-than predicate (a); and predicate engine (b).

entially, this operation retrieves two sets of records and outputs those records that have common attribute values. It is equivalent to a relational equijoin operation whose complexity is about the same as that of a tape-oriented merge of two files. However, in a parallel architecture using a broadcasting LAN, we need a unique algorithm.

The RETRIEVE COMMON algorithm for multiple back ends interconnected by a broadcasting LAN is as follows:

- 1) In SQMD mode, each back end retrieves its first subset of records on the basis of the first query given.
- 2) For each record retrieved, each back end identifies the common attribute specified in the primary operation, hashes its attribute value into a virtual memory address, stores the record in the virtual memory so addressed, and repeats this step until all the records retrieved have been hashed into the back end's virtual memory.
- 3) In SQMD mode, each back end then retrieves the second subset of records on the basis of the second query given.
- 4) For each record retrieved, each back end identifies the

common attribute as specified in the primary operation; hashes its attribute value into a virtual memory address; fetches all records with the same virtual address, if any, from the virtual memory; compares its attribute value with the attribute value of the records fetched; outputs both records when they do compare; and repeats this step until all the records of the second subset have been retrieved and processed.

- 5) Each back end then broadcasts its second subset of records to all the other back ends.
- 6) For each record received via broadcasting, each back end repeats step 4. When step 4 is finally completed for all records, the operation ends.

We must note that, although there are two sets involved in the operation, only the second set was broadcast. If there are n back ends, we have, at most, n mutually exclusive subsets of the second set, each of which is broadcast to the other $(n - 1)$ back ends. The load of the broadcast LAN is therefore the cardinality of the second set, that is, the sum of the cardinalities of its subsets.

MDBS performance

We measure MDBS performance in two ways: response time reduction, or RTR, and response time invariance, or RTI.

If the response time of a transaction is too long for our liking, we may add more database back ends, replicate both the system software and metadata on the new hardware, and redistribute the same base data onto existing and new stores. Ideally, the reduction in response time is inversely proportional to the hardware added. For example, if we double the number of back ends for a transaction in a database, we would like a reduction in response time of one half. This is the RTR.

If the response time gets longer as the database grows in size, we may add more hardware, replicate the software and metadata as above, and redistribute the increased base data onto the existing and new hardware. Ideally, the response time of the transaction will remain the same as long as the new hardware added to the configuration is in proportion to the growth of the database, say, a doubling of the number of back ends for every doubling of database size. This is the RTI. See Demurjian¹² for formal definitions and formulas of RTR and RTI.

Benchmark transactions. As an example, let's use a set of benchmark transactions and a sizeable benchmark database to illustrate RTR measurements. The database is redistributed on a variety of back-end configurations, ranging from the baseline configuration (one-back-end), to a highly parallel configuration (eight-back-end). (Refer to Figure 1.)

Let's also introduce a set of benchmark databases whose sizes are proportional to storage capacities and the number of back ends. We use the set of benchmark databases and the same set of benchmark transactions to conduct RTI measurements over eight configurations as well. There are two kinds of benchmark transactions: overhead-intensive and data-intensive.

Overhead-intensive transaction. This transaction tends to access a small portion of a database and perform intended database primary operations on even smaller portions of accessed data. Let's say that an overhead-intensive transaction (Transaction 1) is a RETRIEVE operation that accesses only 4 percent of the benchmark database, checks them against the query, and outputs only those satisfying the query. The output is about one half the data accessed, or 2 percent of the database. Let's use another overhead-intensive transaction (Transaction 6), a DELETE operation. Like Transaction 1, it accesses 4 percent of the database and qualifies one half, or 2 percent of the data accessed. Unlike Transaction 1, though, it deletes those qualified 2 percent from the database.

Data-intensive transaction. This transaction tends to access a large portion of the database and perform the intended operation on a relatively small portion of the accessed data. Let's consider three retrieves that are data-intensive. Transaction 2 accesses 26 percent of the database,

of which 96 percent of the records (25 percent of the database) satisfy the query. Transaction 3 accesses 50 percent of the database, of which one half of the records (25 percent of the database) satisfy the query. Transaction 4 accesses 100 percent of the database, of which one half of the records (50 percent of the database) satisfy the query. Two DELETES are also data-intensive. Transaction 5 accesses 50 percent of the database, with one half of the records (25 percent of the database) being deleted. Transaction 7 accesses 100 percent of the database, with one half of the records (50 percent of the database) being deleted.

We have now proposed seven benchmark transactions, where two are overhead-intensive and five are data-intensive. (Typical database operations are mostly data-intensive.) We have restricted these seven transactions to two primary database operations, RETRIEVE and DELETE. The three remaining primary database operations, INSERT, UPDATE, and RETRIEVE COMMON, are not considered for benchmarking.

Because INSERT is not a set operation, it cannot take advantage of the parallelism of our database computer. There should be no response time reduction or response time invariance over the conventional sequential database computer.

UPDATE can be viewed as a series of three operations: RETRIEVE a set of records for updating, DELETE the original set of records from the database stores, and INSERT one at a time all newly updated records.

Because RETRIEVE and DELETE have been included in the benchmark study, and INSERT has been left out, there is no need to benchmark the update function.

Finally, we have also decided not to benchmark RETRIEVE COMMON, which is not only a set operation but also involves two sets of records. At present, RETRIEVE COMMON works well with two small sets of records. They are too small to accommodate the sizes of our two benchmark record sets. The size limitations are not attributed to the reliability of the broadcast protocols, nor are they attributed to the parallel and broadcast algorithms of RETRIEVE COMMON outlined above. They are attributed to the microprocessor-based backplanes of the Ethernet. We discuss hardware issues later.

Benchmark databases. The choice of database, cluster, and record sizes for our benchmark study is the first step toward the creation of the benchmark databases.

Multiplicative factor. For the RTR study the same database is benchmarked over one to eight back ends. Thus, the database size must be divisible by each back-end configuration. We must choose, therefore, the lowest common multiple of the back-end numbers as the factor for the database size. Thus, the lowest common multiple of 1, 2, 3, 4, 5, 6, 7, and 8 is 840. For the RTI study we replicate the same database on additional back ends. Thus, for this study the total database size is the number of back ends times the size of a database.

Record sizes. There are three record sizes: 1,000 bytes for a large record, 500 bytes for a medium-size record, and 100

bytes for a small-size record. There are three attribute types: one Type A and two Type Bs. The attribute values take only a small number of bytes in each record. The rest of the record consists of filler attribute values whose attributes are not typed or kept in the metadata tables. The value ranges of the Type A attribute and the distinct values of the two Type B attributes are carefully chosen to induce the following cluster sizes.

Cluster sizes. For convenience in our benchmark effort, we chose the same byte size for each of the four different cluster sizes, although the record numbers in these clusters may be different. The four cluster sizes are large (1,000-byte records), medium-large (500-byte records), medium (200-byte records), and small (100-byte records). Each cluster contains 1.68 Mbytes.

More specifically, there are 140 large clusters, ranging from four 1,000-byte records per cluster to twenty 1,000-byte records per cluster. Together, these 140 clusters contain 1,680 large records, although some contain as few as four large records each, and others contain as many as 20 records each. Similarly, 140 medium-large clusters range from eight 500-byte records per cluster to forty 500-byte records per cluster. The medium-large clusters contain, collectively, 3,360 medium-large records. There are 140 medium clusters, ranging from twenty 200-byte records per cluster to one hundred 200-byte records per cluster. The medium clusters collectively contain 8,400 medium records. Lastly, 140 small clusters range from forty 100-byte records per cluster to two hundred 100-byte records per cluster. The small clusters contain, collectively, 16,800 small records.

Database sizes. The size of the database is the sum of all the records in all the clusters (4×1.68 Mbytes = 6.72 Mbytes). For the RTR study, this figure must be a multiple of the factor 840. Since 6.72 Mbytes = $840 \times 8,000$ bytes, we can redistribute the same 6.72-Mbyte database evenly over two-, three-, four-, five-, six-, seven-, or eight-back-end configurations.

For the RTI study, we simply replicate the 6.72-Mbyte database n times for any n -back-end configuration. Thus, for the eight-back-end configuration, the benchmark database is 53.76 Mbytes (6.72 Mbytes $\times 8$).

Benchmark results. We performed separate RTR studies on all eight configurations by redistributing the same database for the five data-intensive and two overhead-intensive transactions, (minimally, eight database loads for 56 transaction runs). We summarize the results in Figure 6 (p. 56). Since there are seven benchmark transactions, there are seven RTR graphs in Figure 6. We have also completed separate RTI studies on eight configurations by replicating the database seven times for the five data-intensive and two overhead-intensive transactions (minimally, another eight database loads and another 56 transaction runs). Figure 7 on p. 57 shows the RTI statistics and graphs.

RTR results on data-intensive operations. Figure 6a shows the graph for Transaction 2, which accesses 96 percent of the

30,240 records in the 6.72-Mbyte database and selects 26 percent (7,560 records of various sizes) of the records retrieved as its output. It takes about 12 seconds to complete the RETRIEVE operation in the baseline configuration. As the same database is distributed to the two-back-end configuration evenly, we expect the ideal response time for Transaction 2 to be about 6 seconds. Since we doubled the back-end number, we expect the response time to be cut in half. Instead, the measured response time is about 7 seconds. Thus, the overhead incurred with two parallel back ends is about 1 second.

As we increase the back-end number and redistribute the same database, correspondingly, at one third each, one fourth each, one fifth each, one sixth each, one seventh each, and one eighth each for each new configuration, the measured performance curve continues to slide in Figure 6a. This indicates that the additional back ends cut down response time. Furthermore, the differences of the measured response time and the corresponding ideal response time for all configurations remain the same, about 1 second. In other words, this kind of parallelism does not increase the overhead.

In Figure 6b for Transaction 3, Figure 6c for Transaction 4, Figure 6d for Transaction 5, and Figure 6e for Transaction 7, the plotted curves all slide downward. The plotted curves again closely parallel the ideal curves, regardless of the number of back ends configured. These figures indicate that use of a multiplicity of back ends to reduce the response time of data-intensive operations is a promising approach. The curve has not yet leveled off at the eight parallel database processor-store pairs. We may approach a higher degree of parallelism beyond eight for more reductions in response time.

RTR results on overhead-intensive operations. In Figures 6f and 6g, we show the measured RTR statistics on Transactions 1 and 6. As overhead-intensive operations, these two transactions access only a small amount (248.8 Kbytes) of records data (4 percent of the 6.72-Mbyte database) from the database stores. However, they access a relatively large amount of in-memory processing (50 percent of the records retrieved for qualification or deletion). As we can see from the graphs, the curves level off at the five-back-end configuration. Thus, for overhead-intensive operations, there is no appreciable reduction in response time beyond this degree of parallelism. In other words, overhead-intensive operations do not require a highly parallel database computer. Instead, we should improve the performance of individual back ends with types such as RISC processors.

RTI data-intensive operations. In Figure 7a, we see that all five data-intensive operations, Transactions 2, 3, 4, 5, and 7, exhibit relatively level curves in all eight configurations, that is, from the baseline to the eight-back-end configuration. This indicates the response time of each transaction remains unchanged, or invariant, despite a change in database size from the baseline 6.72 Mbytes to 53.76 Mbytes in the eight-back-end configuration. In other words, even though

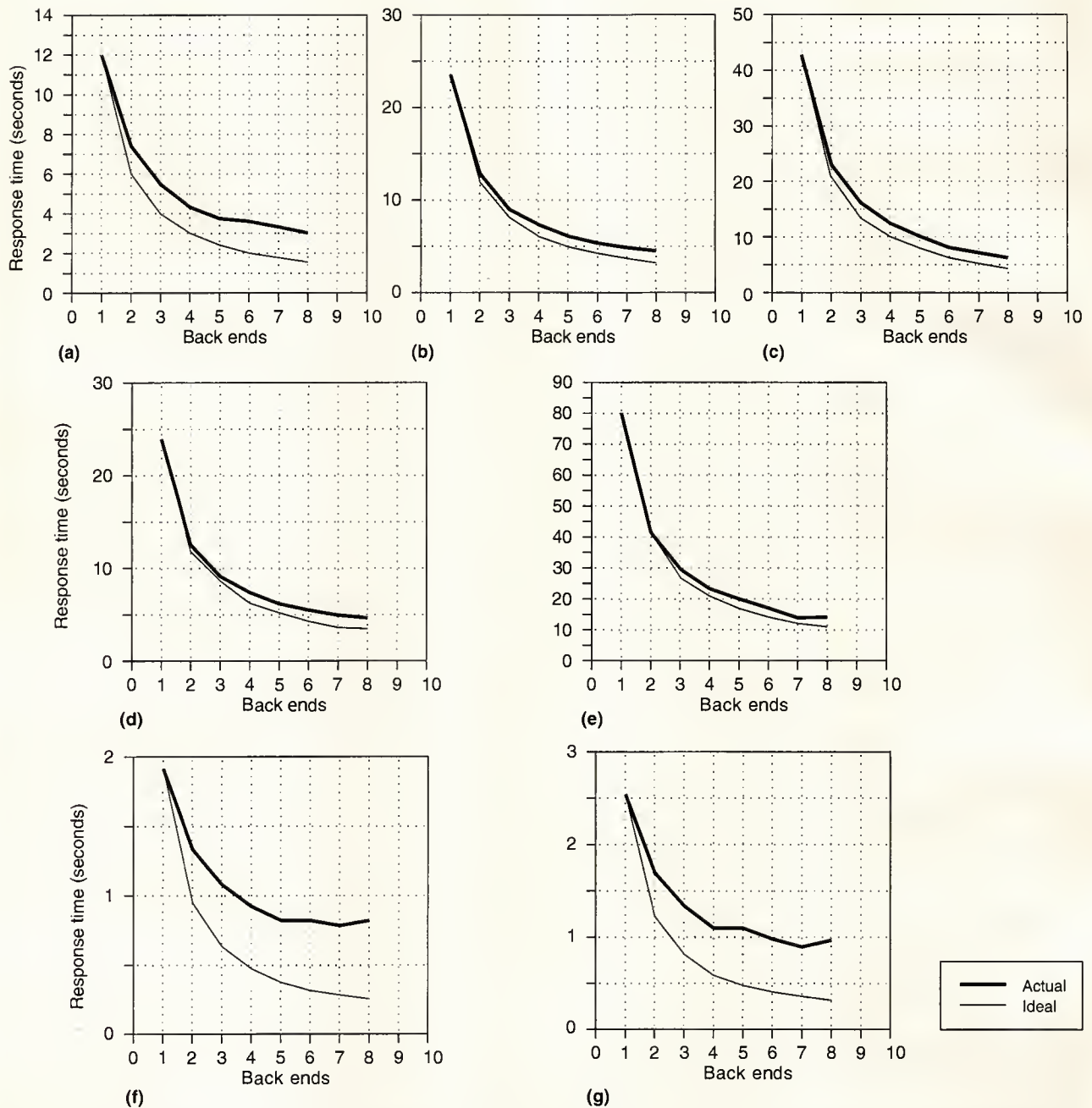


Figure 6. Response-time reductions over the multiplicity of parallel back ends for transactions 2 (a), 3 (b), 4 (c), 5 (d), 7 (e), 1 (f), and 6 (g).

the transaction must operate on more data, the response time of the transaction remains invariant, as long as the multiplicity of the parallel back ends is proportional to the increase in database size.

Transaction 7 (a DELETE operation) almost doubles the response time of Transaction 4 in every configuration. This is not surprising. Both access the same amount of data (the entire database). However, Transaction 7 must not only

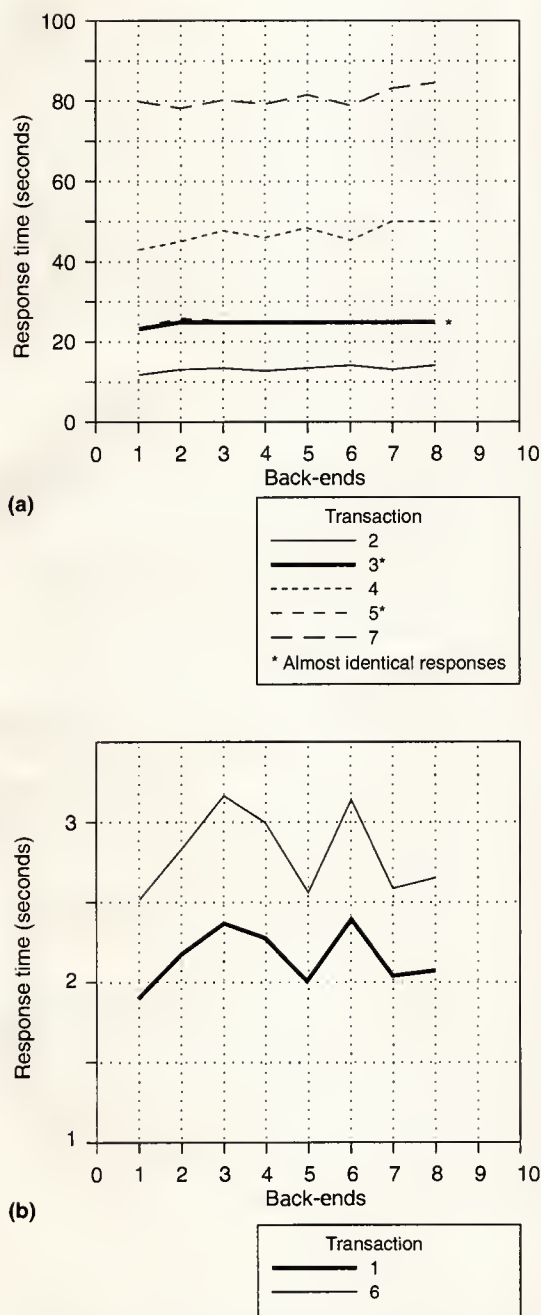


Figure 7. Response-time invariances over the multiplicity of parallel back ends for data intensive transactions (a) and for overhead-intensive transactions (b).

select 50 percent of the records accessed as does Transaction 4 (a RETRIEVE operation) but must also delete the selected records from the database stores. The curves indicate that it

takes some 40 seconds for each back end to access and process 30,240 records of varying sizes, and it takes some 30 seconds more for each back end to tag 15,120 records for deletion and return them to their database stores for later removal.

Transaction 5 (another DELETE operation) and Transaction 3 (a RETRIEVE operation) have almost identical response times for all configurations. This is also not surprising, since they both access the same half of the database and select half of the records retrieved (for deletion in the case of Transaction 5, or for returning data in the case of Transaction 3). But, when the amount of accessing is relatively small and the amount of processing is again relatively small, the times for writing the smaller number of deletion tags onto database stores are mostly overlapped by the accessing and processing times of the other records.

RTI results on overhead-intensive operations. Figure 7b contains the response times for the overhead-intensive operations in Transactions 1 and 6. Both curves exhibit a similar zigzag pattern. This indicates that, as the benchmark database is doubled, tripled, quadrupled, quintupled, sextupled, septupled, and octupled for the different configurations, the data has not been evenly redistributed on the corresponding configurations. The back end that has the highest number of records in a multiback-end configuration has the longest response time. Nevertheless, the deviation is within only one-half second in Transaction 1, and within 1 second in Transaction 6. It is important that, for either transaction, the response times deviate upward and downward from the norm. The norm for Transaction 1 is about 2.25 seconds. The norm for Transaction 6 is about 2.75 seconds.

MDBS limitations

There are several hardware and software limitations, but the prospects of MDBS-like database computers will be great if we can overcome the limitations.

Hardware. The hardware limitations seem to be related to the microprocessor-based Ethernet. A dual-ported, 124-Kbyte real memory supports two microprocessors in a back-plane. This real memory also buffers the messages coming from and going to the Ethernet cable. Since messages are typically small and infrequent, the real memory size is adequate for buffering messages. Further, messages are sent mostly via the point-to-point communication protocol, which cannot cause a collision.

For buffering records, though, the size of the real memory is inadequate, since records are individually large, typically hundreds of bytes per record, and are transmitted in bulk, typically tens, if not hundreds, of records in a batch transmission. The buffer, therefore, overflows, and some of the records are overwritten by records arriving later. This requires the sender to repeat the send protocol containing the last batch of records. Repetitions of this type markedly slow the broad-

cast operations in the second stage of a RETRIEVE COMMON operation where each back end broadcasts its subset of the second set of records to all the other back ends. As one or more back ends must repeat the broadcast operations, the delay and traffic become intolerable. This, of course, severely slows RETRIEVE COMMON.

Larger buffer memories and interrupt mechanisms. We make two recommendations for the purpose of overcoming this hardware limitation: provide a larger, dual-ported buffer memory in each backplane and provide an interrupt mechanism. Whenever the buffer memory is full, the backplane microprocessors interrupt the CPU (in our case, the back-end computer's 68020 microprocessor) so the CPU can begin the Get process to receive the records in the buffer right away. Without this interrupt hardware, there is no quick way to alert the back-end computer's CPU when the buffer is full and ready to forward the records. Meanwhile, Get, the sole process for receiving records or messages from the Ethernet for other processes, may be put on standby, while the CPU executes another process.

Higher bandwidth communication network. The Ethernet has a bandwidth of 10 Mbps. This is adequate for all the database primary operations, except RETRIEVE COMMON, where the bandwidth must sustain the cardinality of the second set of records in the broadcast mode. Say the cardinality is 25,000 records, and the average record size is 750 bytes. Then eight subsets of the second set of records of 18.75 Mbytes are broadcast at about the same time over the Ethernet. Not counting the delay due to eightfold collisions during the period of multibroadcasting, the present bandwidth requires at least 15 seconds to complete the transmission. Assuming that it takes 2 seconds to resend one eighth of the record set when a collision occurs, a minimum of 16 seconds are needed for the second set. This is also too slow.

Software limitations. Software limitations are related mainly to INSERT, UPDATE, and RETRIEVE COMMON, three of the five database primary operations.

Massive loading of a new database. For our benchmark effort, we created benchmark databases. Each database is a maximum 6.72 Mbytes per back end. We used INSERT to create each database by adding the records one at a time. This process is, of course, very time-consuming. It took us days to create one database and weeks to create all the necessary databases. We needed a massive loading program that could read the attribute-value pairs generated from a record generator, format them into records, extract their typed attributes, form their attribute table, and construct the descriptors from the typed attributes. We also needed it to form the descriptor table, determine the clusters of all records, form their cluster table, load the attribute, descriptor, and cluster tables onto the metadata disks, and load clustered records evenly onto base data disks one cluster of records at a time.

Control of virtual-memory space. In both UPDATE and

RETRIEVE COMMON, each back end must first select a set of records to be processed. Regardless of its size, this record set arrives from the database stores and temporarily resides in the virtual memory, awaiting subsequent operations. Typically, we use hashing techniques to quickly come up with addresses for such temporary storage. Unfortunately, the addresses are virtual—not real—addresses. Consequently, some or nearly all of the records in the temporary storage page out of real memory. It takes many paging (I/O) operations to retrieve the records for the next processing step in real memory. This delays or slows down both the UPDATE and RETRIEVE COMMON operations.

We make the following recommendations here for the purpose of overcoming this limitation:

- The operating system provides us with a system function that enables us to lock a certain number of virtual memory pages into real memory. So, as long as we use these locked pages for the temporary storage of our records, there will be no paging operations involved.
- On the basis of the size of available locked pages, we can partition either an UPDATE or a RETRIEVE COMMON into a series of updates (in which each update works on a new batch of records in the locked pages) or a series of RETRIEVE COMMONS. For example, in the first set of records each RETRIEVE COMMON works on a new batch of records in the locked pages. In other words, each batch of the first record set is processed against the entire second record set, although the second record set can also be brought in for processing one batch at a time (as long as both batches can be accommodated in the available locked pages).

System process priority. The Get process is a very high-priority process that should be constantly listening over the communication net and anticipating messages, data, or transactions heading its way. It is akin to a real-time process. In other words, the CPU of the back end should never deactivate a Get. The operating system should provide a system function that enables us to set the priority of one of our processes with the capability of real time.

DESPITE THE LIMITATIONS the prospects for a microprocessor-based, multiback-end database computer, such as MDBS, are good. These prospects are rooted in the following categories: external technology and intrinsic architecture.

We note several technology prospects.

- 1) All the hardware and software limitations discussed previously can be overcome with present computer technology. We do not have to wait for any distant technology for solutions.

Project support

The work reported here is now supported by funds from the Naval Research Laboratory and the Naval Postgraduate School. It began in 1980 with a proposal for an equipment grant submitted by the author to Digital Equipment Corp. In 1981 DEC and the Office of Naval Research funded the initial MDBS with a VAX 11/780 as the controller computer, two PDP 11/44s and their disk drives as back ends, and some PCLs as the communication network. The use of the 11/780 for the development of MDBS software was the main reason that we had also employed the VAX unit as the controller. This was before the advent of the microprocessor and the arrival of the microprocessor-based Ethernet. All three of our computers were minicomputers, with the VAX using the VMS operating system, and the PDP, the RSX operating system. The Ethernet functional specification had just been announced. We used three PCL cables to "emulate" the Ethernet specification. The two-back-end database computer was operated in the Laboratory for Database Systems Research at Ohio State University.

The MDBS software requirements were largely the work of J. Menon,¹ D. Kerr, A. Orooji, and S. Demurjian accomplished the principal supervision and development.²⁻⁴

The process structure is essentially the same to this day, although we have updated the original code many times. We wrote all processes in C, with five of them running in VMS and the other five in RSX. Get and Put in VMS communicate with Put and Get in RSX. The Office of Naval Research supported our software development effort as well.

In 1983, the Laboratory for Database Systems Research moved to the Computer Science Department of the Naval Postgraduate School. We gave MDBS's VAX 11/780 to Ohio State University and replaced it with one of the two Computer Science Department's VAX 11/780s. In 1985, we began to phase out and replace all the minicomputers and their communications network with microprocessor-based workstations and the Ethernet. We also began to add as many back ends as funds permitted. After we reached the round number of eight, we began to upgrade the microprocessors from 68010 to 68020 chips, and the real memories in both size and speed. The present configuration⁵ is depicted in Figure 1. We recently acquired two Sun-4 workstations with RISC-based microprocessors. However, we have not decided whether we should replace all the eight back ends with Sun-4s or simply add these two to the existing configuration for a heterogeneous, 10 back-end configuration. The replacement and upgrade were funded by the Naval Postgraduate School equipment program and the Department of Defense STARS (Software Technology for Adaptable, Reliable Systems) program.

MDBS at one time was known as MBDS. However, because of a trademark conflict, we changed the name of our database computer to its present name. MDBS is used as a research vehicle for the study of design analyses⁶⁻⁸ and performance evaluations⁹⁻¹¹ and their methodologies.¹² We also use it to study the database system architectures of the future.¹³⁻¹⁷

- 2) All the back-end hardware requirements for a database computer require no special-purpose computer components or devices. They can be met with off-the-shelf microprocessor-based hardware, such as Sun-4 workstations or super PCs, each of which can support several hard disks of various capacities.
- 3) The LAN hardware requirements for a database computer can also be met with an Ethernet-like LAN with wider bandwidths, higher transmission rates, and reliable multibroadcast capabilities. Such LANs can be made available commercially.
- 4) Microprocessor technology is progressing at a higher rate than that of both minicomputers and mainframes. We should be able to ride on the success of these faster advances.
- 5) In storage technology, Winchester-type disks and disk arrays based on that principle are improving more rapidly than are the densities and capacities of standard disk drives. By relying on the gains in Winchester-type

drive technology, we should have a better mix and choice of disk drives with different capacities and numbers for separate metadata, base data, and page stores.

We also note some architectural prospects.

- 1) Parallel architecture uses identical back ends. Also, database back-end software and metadata are replicated. Only base data require redistribution. Thus, the cost of any additional hardware and software is minimized. It is rather straightforward to increase parallelism readily and cost-effectively.
- 2) An increase in parallelism can be scaled for the purpose of achieving the desired response time for a given transaction and/or a certain capacity of a database.
- 3) The increase in performance for a given response time reduction or a capacity increase for a response time invariance is directly proportional to the number of back ends and corresponding stores added.

The MDBS architecture is the most effective and efficient one in reducing the response time of a transaction. It is also the most effective and efficient architecture to maintain the same response time of a transaction, despite the increase in size of a database. ■

Acknowledgments

The MDBS software development involved a large number of undergraduates, graduates, postgraduates, professors, and other professionals. They cannot all be named here without committing omissions. However, D. Kerr, A. Orooji, and S. Demurjian accomplished the principal supervision and development.

References

1. S.A. Demurjian, D.K. Hsiao, and J. Menon, "A Multiback-End Database System for Performance Gains, Capacity Growth, and Hardware Update," *Proc. Second Int'l Conf. Data Engineering*, IEEE CS Press, Los Alamitos, Calif., 1986.
2. D.K. Hsiao et al., "The Implementation of a Multiback-end Database System (MBDS): Part I, An Exercise in Database Software Engineering," *Advanced Database Machine Architecture*, Ch. 10, Prentice-Hall, Englewood Cliffs, N.J., 1983.
3. H. He et al., "The Implementation of a Multiback-end Database System (MBDS): Part II, The Design of a Prototype MBDS," *Advanced Database Machine Architecture*, Prentice-Hall, 1983, Ch. 12.
4. R.D. Boyne et al., "A Message-Oriented Implementation of a Multiback-end Database System (MBDS)," *Database Machines*, H. Leilich and M. Missikoff, eds., Springer-Verlag, New York, 1983.
5. D.K. Hsiao, "Super Database Computers: Hardware and Software Solutions for Efficient Processing of Very Large Databases," *Proc. IFIP 86 Congress*, IFIP Press, Geneva, 1989.
6. D.K. Hsiao, "Cost-Effective Ways of Improving Database Computer Performance," *AFIPS Conf. Proc.*, Vol. 52, AFIPS Press, Reston, Va., 1983.
7. D.K. Hsiao and P. Strawser, "The Predicate Machine—A High-Level Database Computer," *Proc. Int'l Conf. High-Level Language Computer Architecture*, 1984.
8. D.K. Hsiao, "The Impact of the Interconnecting Network on Parallel Database Computers," *Database Machines*, H. Tanaka and M. Kisuregawa, eds., Kluwer Academic, Boston, 1987.
9. S.A. Demurjian et al., "Performance Evaluation of a Database System in Multiback-end Configurations," *Database Machines*, D.J. DeWitt and H. Borel, eds., Springer-Verlag, 1985.
10. S.A. Demurjian et al., "A Computer-Aided Benchmarking System for Parallel and Expandable Database Computers," *Proc. 1987 Fall Joint Computer Conf.*, AFIPS Press, 1987.
11. J.E. Hall, D.K. Hsiao, and M. Kamel, "Performance Evaluations of a Parallel and Scalable Database Computer," *Proc. Conf. Databases, Parallel Architectures, and Their Applications*, N. Rishe, ed., IEEE CS Press, 1990.
12. S.A. Demurjian, D.K. Hsiao, and R. Marshall, *Design Analysis and Performance Evaluation Methodologies for Database Computers*, Prentice-Hall, 1987.
13. D.K. Hsiao, "Future Database Machine Architectures," *New Directions for Database Systems*, Ch. 2, R. Ariav and A. Clifford, eds, Ablex Publishing Corp., Norwood, N.J., 1986.
14. S.A. Demurjian and D.K. Hsiao, "Towards a Better Understanding of Data Models through the Multilingual Database System," *IEEE Trans. on Software Engineering*, Vol. 14, No. 7, July 1988, pp. 946-950.
15. D.K. Hsiao and M. Kamel, "Heterogeneous Databases: Proliferations, Issues, and Solutions," *IEEE Trans. Knowledge and Data Engineering*, Vol. 1, No. 1, Mar. 1989, pp. 45-62.
16. D.K. Hsiao, "Databases and Database Systems in the 21st Century," *Very Large Scale Computation in the 21st Century*, Ch. 9, J. Mesirov, ed., SIAM, Philadelphia, 1991.
17. D.K. Hsiao, "Federated Databases and Systems, A Tutorial," (to appear in) *Int'l J. Very Large Data Bases*, Vol. 1, No. 1, Mar. 1992.



David K. Hsiao is a professor of computer science at the Naval Postgraduate School in Monterey, California. His technical interests include database systems and computers, database security and access control, and federated heterogeneous databases and systems.

Hsiao earned BS and MA degrees in mathematics from Miami University and a PhD in computer and information science from the University of Pennsylvania. He became an IEEE Fellow in 1989 and also served as a member of the governing board of the IEEE Computer Society. He was an elected member-at-large of the ACM Council and is a trustee of its Very Large Data Base Endowment.

Address questions concerning this article to David K. Hsiao, Computer Science Department, Naval Postgraduate School, Monterey, CA 93943; or via e-mail at hsiao@cs.nps.navy.mil.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 162	Medium 163	High 164
---------	------------	----------



Rinda:

A Relational Database Processor with Hardware Specialized for Searching and Sorting

This database processor accelerates nonindexed relational database queries. Rinda is composed of content search processors and relational operation accelerating processors; the former search rows stored in disk storage, and the latter sort rows stored in the main memory. The processors connect to a general-purpose host computer with channel interfaces. Performance improves substantially as Rinda executes queries three to 100 times faster than conventional database-management system software in a benchmark system.

Ushio Inoue

Tetsuji Satoh

Haruo Hayami

Hideaki Takeda

Toshio Nakamura

Hideki Fukuoka

NTT Communications

and Information Processing

Laboratories

Various database machine architectures have been proposed and developed to solve the performance problems of relational databases. Database machines fall into two classes on the basis of their functions: computers and processors. Database computers, such as the Britton Lee Intelligent Database Machine and the Teradata DBC/1012, are independent of their host computers and perform all database-management functions by themselves. Database processors, on the other hand, depend on their hosts and perform only certain database functions. For example, CAFS¹ is dedicated to ICL mainframes, and its functions are limited to searching and filtering table rows stored in disk storage. IDP² is an optional processor for Hitachi's largest mainframe, and it can perform only sorting and merging of keys stored in the main memory. In general, database processors achieve higher performance at lower costs, because specialization in their hardware is restricted to database functions that have caused severe performance problems in general-purpose computers.

We can classify relational database access into queries and updates, which we can further classify into nonindexed and indexed queries and updates.³ Database systems execute indexed queries and updates efficiently using indexes created

before the execution. A typical indexed query is a selection of a single row in a table by exact matching of a unique search key. With such queries or updates, current database-management systems perform fairly well. For nonindexed queries, however, systems cannot take advantage of indexes. Examples of nonindexed queries are a selection of multiple rows by partial matching of a character string, a join of two tables with nonunique join keys, and an aggregation of a whole table with groups of rows in it. A general-purpose computer consumes a lot of processing time executing such queries because searching and sorting are computationally intensive operations.⁴

At NTT, we developed a database processor called Rinda to accelerate nonindexed relational database queries.³ Rinda is composed of content search processors and relational operation accelerating processors.⁵ A content search processor, or CSP, searches rows in disk storage and transfers the selected rows to the main memory. A relational operation accelerating processor, or ROP, sorts the rows stored in the main memory. The CSP and ROP perform their work using hardware specialized for searching and sorting. Rinda is connected to a general-purpose host—our DIPS series mainframe^{6,7}—with channel interfaces, and is controlled by a Rinda control program running on the host.

Rinda architecture

In relational database systems, searching and sorting are basic operations. Searches select rows and columns from a table according to predicates in a query and are the first step in query execution. When a table is stored on disk, disk reads are also needed. There are two problems with searches. For nonindexed queries, they take a great amount of CPU time because all rows in a table must be qualified. Transferring all rows from the disk to the main memory also takes extra time. A solution to these problems is an intelligent disk controller that performs searches at the disk storage.

Sorts order rows in a table using a key, which may be composed of several columns. Systems can perform join, nested, and aggregate queries efficiently when rows are sorted. The major problem in sorting is CPU time consumption, because its complexity with ordinary algorithms is $N * \log(N)$, where N is the number of rows. Another problem stems from memory storage because additional disk accesses are required when memory size is insufficient. A solution to these problems is a hardware sorter with a large memory, which sorts rows in time N .

Design considerations. Our goal in developing Rinda was to relieve host computers from heavy loads caused by nonindexed queries. Therefore, Rinda had to satisfy the following requirements:

- 1) It should be a database processor, so the existing host computer and disk storage would not need to be replaced.
- 2) The application program interface should be SQL to avoid any modification of user programs.

To meet the first requirement, Rinda uses the two special-purpose processors just introduced: CSPs and ROPs. A CSP connects to a host computer and a disk controller with channel interfaces, forming an intelligent disk controller. An ROP that uses a hardware sorter also connects to the host by a channel. To meet the second requirement, we based Rinda's functions on a subset of SQL functions. Software on the host computer fills the gaps between full-set SQL and Rinda.

Hardware architecture. Figure 1 shows a typical Rinda system organization. The major components are a DIPS series host computer, standard disk controllers and disk drives, CSPs, and ROPs. Database size and performance requirements determine the number of CSPs and ROPs in a system. For example, if the database is very large, several CSPs used

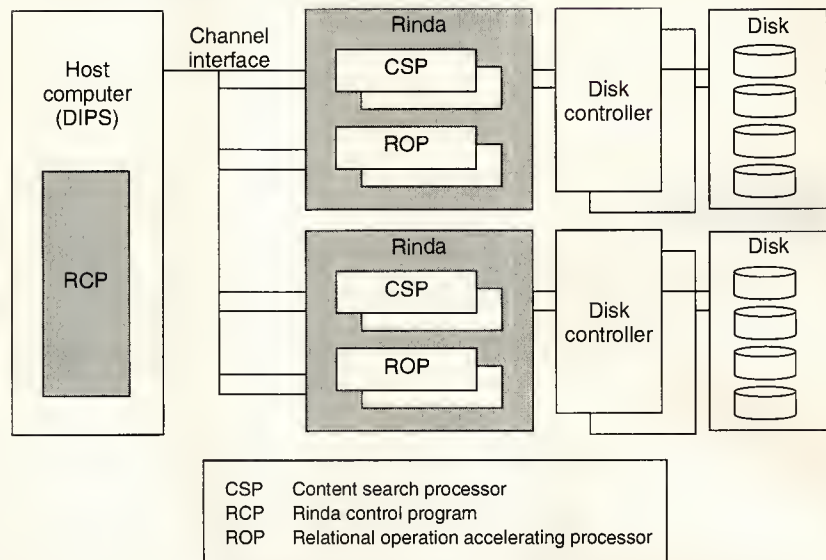


Figure 1. Rinda system organization.

Table 1. Primary CSP functions.

Function	Description
Predicates	Specified in a WHERE clause
Comparison	<column> <comp-op> <value>
In	<column> [NOT] IN <value list>
Like	<column> [NOT] LIKE <pattern>
Null	<column> IS [NOT] NULL
Boolean expression	Any combination of predicates
Column selection	SELECT <column list>
Set function	Count(*)

in parallel reduce the I/O time. Channel commands and order tables created by the Rinda control program, or RCP, and running on the host control the CSPs and ROPs.

A CSP searches a database table stored on a disk, selects rows and columns specified by a query and transfers only the results to the host. Table 1 shows its primary functions, which cover most single-table queries of relational databases. The CSP performs these functions at the disk's data-transfer rate.

An ROP sorts rows in a table transferred from the main memory of the host and transfers the results back to the host. Its primary functions, listed in Table 2, include removal of unnecessary rows to accelerate join and nested queries. The ROP performs these functions at the data-transfer rate of the channel. We discuss CSP and ROP hardware structure in detail later.

Table 2. Primary ROP functions.

Function	Description
Sorting	Order By <column[ASC/DESC] list> (also used for joins, subqueries, and Group-By clauses)
Filtering	Removal of unnecessary rows (used for join and nested queries)
Duplicate removal Set function	DISTINCT <column list> Count(*) with a Group-By clause

Software architecture. Figure 2 shows the host computer's software structure. The Rinda control program is attached to an existing relational database-management system to form a new integrated DBMS. The language-processing subsystem analyzes queries written as SQL statements. If the query is nonindexed, the system calls RCP functions, which optimize and execute queries using Rinda.

Figure 3 illustrates a typical procedure to execute a nonindexed query. First, the RCP sends a command to the CSPs, which transfer selected rows to the RCP buffers in the main memory. Next, the RCP sends another command to an ROP, which sorts the rows in the buffers. Finally, the RCP returns the results to the user program.

The RCP also allows a table to be stored over multiple disks and conducts parallel CSP searching on the disks. The database control subsystem keeps the database consistent when nonindexed queries and indexed updates execute concurrently in multiuser environments.

Related work. Several searching and sorting processors have been developed. For example, CAFS¹ is a searching processor with functions similar to those of the CSP. However, the CSP has properties different from those of other searching processors. First, its design is based on SQL functions, and it supports the null value. Second, it deals with the page format of an existing DBMS so no data conversion is required to introduce Rinda to installed user database systems. As for sorting processors, existing hardware sorters⁸ can sort only a small number of rows at one time because the number of processing elements is limited. The ROP solves this problem with multiway merging. Moreover, we integrated sorting and

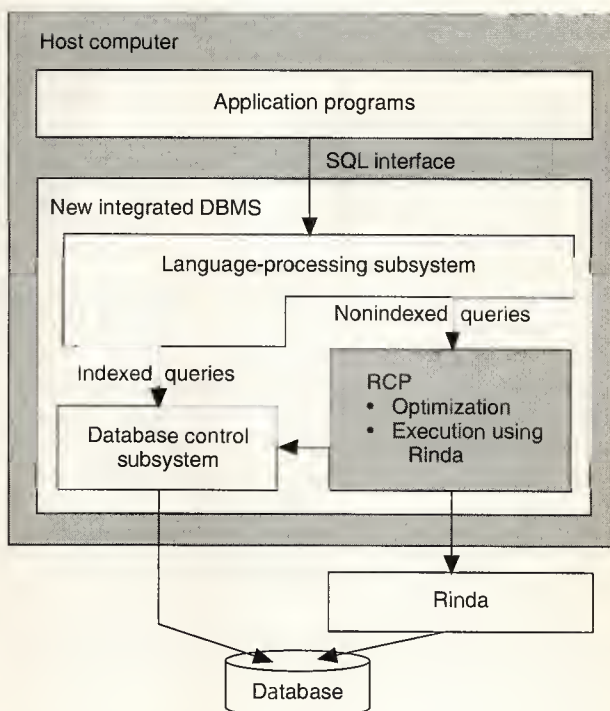


Figure 2. Software structure on a host computer.

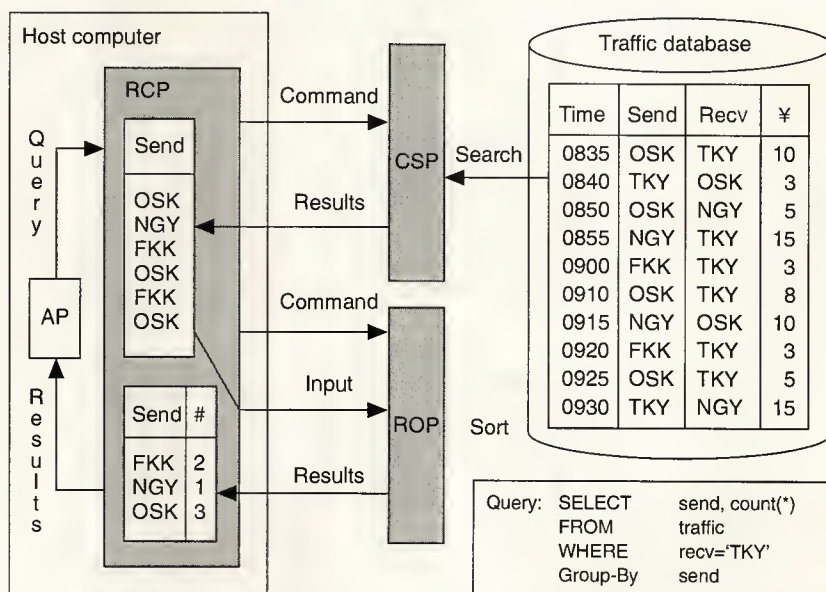


Figure 3. Typical query execution using Rinda (AP indicates the application program).

filtering in the ROP to use the three-phase join method.

Content search processor

A database consists of database spaces, each of which is a collection of pages stored in one or more continuous disk areas. A database space may be distributed over several disks. When a table is created in a database space, adjacent pages are assigned to the table. As the table grows, a constant addition of adjacent pages increases the table. Thus, a table is stored in multiple disk extents. Each page holds rows of a single table, and no row is stored across page boundaries. Each table contains control records indicating its disk extents in separate pages, and the RCP uses this information to perform CSP searches.

CSP organization. A CSP search can be performed in two ways, synchronously and asynchronously. In the synchronous mode, a data stream from a disk processes on the fly, while in the asynchronous mode, buffers decouple reading and searching. The CSP performs only asynchronous searches for the following reasons:

- A disk controller has an error-detecting and -correcting facility for data pages. This facility completes after the last byte of the page transfers from the disk to the buffer.
- We had designed the page format without any consideration of hardware-searching mechanisms. Changing the page format from software-oriented to hardware-oriented was impossible because it would require database conversion for existing systems.

The CSP transfers pages successively from a disk to its buffers using the multitrack-read method and scans the page in the buffer before the next page comes. Therefore, the CSP performs a search within the page-transfer time, similar to doing it on the fly. Figure 4 shows a block diagram of the CSP.

Search operation flow. Rinda systems perform searches as follows:

- 1) The RCP running on the host acquires the RCP buffers and prepares a set of channel commands and an order table for each CSP. An order table contains disk access information that includes the disk unit number, extent addresses to be searched, the RCP buffer size, and a search condition.
- 2) Channel commands send the order tables from the host to the CSPs.
- 3) According to the order table, each CSP generates new channel commands to the disk to read pages successively using the multitrack-read method. When the first page arrives, the CSP begins to select rows from the page. To

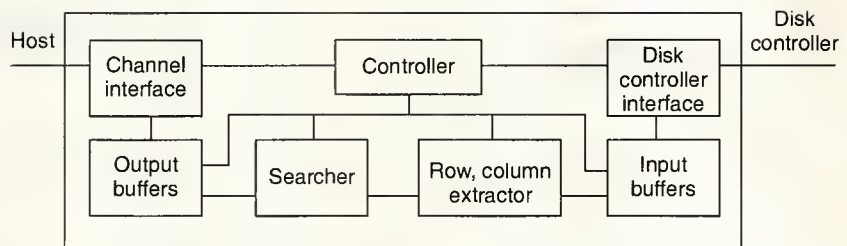


Figure 4. CSP block diagram.

keep up the data-transfer rate, the CSP uses multiple input buffers that hold pages transferred from the disk, and multiple output buffers that hold new pages containing selected rows.

- 4) When one of the output buffers fills, the CSP transfers the page to the host asynchronously.

This procedure continues without any interruption to the host until the CSP has read all the extents or until all the RCP buffers are occupied. In the latter case, the RCP saves the selected rows on a work disk and repeats the search procedure.

Evaluation of search conditions. A search condition is expressed by predicates and Boolean operators And, Or, and Not. In SQL, when the value is undefined, each column may have a null value instead of zero or space. As a result, the truth value of a predicate in the condition may be true, false, or unknown. Therefore, the three-valued logic (true, false, and unknown) shown in Figure 5, instead of two-valued logic (true and false), defines Boolean operators.

And	True	False	Unknown
True	True	False	Unknown
False	False	False	False
Unknown	Unknown	False	Unknown

Or	True	False	Unknown
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

Not	True	False	Unknown
	False	True	Unknown

Figure 5. Truth tables of the three-valued logic.



December 1991 issue (card void after June 1992)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest

(Add comments on the back)

Readers, indicate your interest in articles and departments by circling the appropriate number (shown on the last page of articles and departments):

150 151 152 180 181 182
153 154 155 183 184 185
156 157 158 186 187 188

159 160 161 189 190 191
162 163 164 192 193 194
165 166 167 195 196 197

168 169 170 198 199 200
171 172 173 201 202 203
174 175 176 204 205 206

177 178 179 207 208 209

Product Information 1

(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	



December 1991 issue (card void after June 1992)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest

(Add comments on the back)

Readers, indicate your interest in articles and departments by circling the appropriate number (shown on the last page of articles and departments):

150 151 152 180 181 182
153 154 155 183 184 185
156 157 158 186 187 188

159 160 161 189 190 191
162 163 164 192 193 194
165 166 167 195 196 197

168 169 170 198 199 200
171 172 173 201 202 203
174 175 176 204 205 206

177 178 179 207 208 209

Product Information 2

(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	

SUBSCRIBE TO IEEE MICRO

All the facts about today's chips and systems

☐ **YES**, sign me up!

If you are a member of the Computer Society or any other IEEE society, pay the member rate of only \$21 for a year's subscription (six issues).

Society: _____

IEEE membership no: _____

Society members: Subscriptions are annualized. For orders submitted March through August, pay half the full-year rate (\$10.50) for three bimonthly issues.

Full Signature _____ Date _____

Name _____

Street _____

City _____

State/Country _____ ZIP/Postal Code _____

☐ **YES**, sign me up!

If you are a member of ACM, ACS, BCS, IEE (UK), IEEE but not a member of an IEEE society, IECEI, IPSI, NSPE, SCS, or other professional society, pay the sister-society rate of only \$38 for a year's subscription (six issues).

Organization: _____

Membership no: _____

☐ Payment enclosed Residents of CA, DC, Canada, and Belgium add applicable sales tax.

☐ Charge to ☐ Visa ☐ MasterCard ☐ American Express

Charge-card number _____

Expiration date _____

Month _____ Year _____

Prices valid through 12/31/91
12/91 MICRO

Charge orders also taken by phone:
(714) 821-8380 8 a.m. to 5 p.m. Pacific time
Circulation Dept.
10662 Los Vaqueros Circ., PO Box 3014
Los Alamitos, CA 90720-1264

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

Reviewers Needed. If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.

For reader-service inquiries, see other side.

PO Box is for reader-service cards only.

PLACE
POSTAGE
HERE

IEEE Micro

PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA



Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

Reviewers Needed. If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.

For reader-service inquiries, see other side.

PO Box is for reader-service cards only.

PLACE
POSTAGE
HERE

IEEE Micro

PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

IEEE COMPUTER SOCIETY

PO BOX 3014
LOS ALAMITOS CA 90720-9804
USA

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



If unknown is replaced by false in the evaluation of predicates, the three-valued logic is localized into two-valued logic, and the CSP implements simply. A where clause in an SQL statement, WHERE <search condition>, selects rows whose final truth value of the <search condition> is true. Therefore, unknown can be replaced by false in the last stage of the evaluation. In the truth table shown in Figure 5, the Not(unknown) column is important. If unknown is simply replaced by false, the result may be unreasonable because Not(false) is true while Not(unknown) is false. Therefore, if there is no Not operator in the search condition, unknown can be replaced by false, and the three-valued logic can be localized into two-valued logic.

The following procedure transforms any search condition to another search condition having the same effect without Not operators:

- 1) Transform a search condition to a conjunctive canonical form.
- 2) If there is no predicate with Not, the procedure ends.
- 3) If there is a predicate with Not, remove Not in the predicate and reverse the comparison operator in the predicate. For example, Not(column1 > x) is replaced by (column1 ≤ x).

In Rinda systems, the RCP transforms a search condition to a conjunctive canonical form without Not operators, and creates an order table for the CSP. The CSP replaces unknown with false in the evaluation of predicates and applies two-valued logic.

Accelerating processors

The purpose of the ROP is to accelerate join queries. We developed a new join algorithm, which makes the best of a hardware sorter.

Join algorithms. There are three principal types of conventional join algorithms: nested-loop,⁹ sort-merge,⁹ and hash¹⁰ algorithms. Nested-loop join algorithms repeatedly compare each row in the outer table with all rows in the inner table. This algorithm is practical only if both tables are small, because it requires a very large number of comparisons. Hash join algorithms split both source tables into several buckets using a hashing function and execute comparisons in each bucket when every row has the same hashing value. Therefore, hash join algorithms are suitable for parallel processors. Sort-merge join algorithms sort both tables to decrease the amount of merge-join computations to a linear order. We selected a sort-merge join algorithm for Rinda because specialized hardware can rapidly execute sorting.

Rinda's three-phase join method^{4,11} based on the sort-merge algorithm consists of filtering, sorting, and merge-join phases. Most unnecessary rows are removed in the filtering phase. Remaining rows, each of which is a candidate of the join, are

sorted in the sorting phase. After this, sorted rows are merged together in the merge-join phase.

The filtering phase uses hashed bit arrays¹² to remove unnecessary rows. Of the three phases, the filtering phase involves the most rows. Therefore, we decided that specialized hardware should execute this step. Sorting consumes much CPU time, so we chose specialized hardware to execute the sorting step. Filtering decreases the number of rows, and the remaining rows are sorted during the sorting phase. Therefore, the RCP on the host executes the merge-join phase with little computation.

ROP organization. Figure 6 shows a block diagram of the ROP, which performs filtering and sorting. Its main functional blocks are the hasher and sorter. The key extractor and row transfer blocks were implemented for a row-level pipeline.

Rows are generally composed of several columns with different data types such as integer, decimal, and character string. A null value may also appear in a column. The ROP has key-extractor hardware to compose a fixed-length and directly comparable internal key from a row. Using the internal keys, simple hardware can rapidly execute filtering and sorting.

The ROP has a single memory for storing internal keys, rows, and hashed bit arrays. The ROP moves the boundaries between these areas dynamically when it stores keys and rows. The constant ratio of the storage capacity determines the size of the bit array to keep the loading factor low. For example, if storage capacity increases, so does the size of the bit arrays. Keys and rows are stored in the remaining ROP memory.

Filtering block. Since unnecessary rows remain from collisions of hashing-function values, we needed a sophisticated hashing function to decrease the number of collisions. An ideal hashing function would distribute all keys uniformly over the addressing space of a hashed bit array. Lum, Yuen, and Dodd¹³ and Knott¹⁴ compared division, multiplication, folding, and other hashing functions on the basis of the number of collisions, using fixed-length short keys. The division

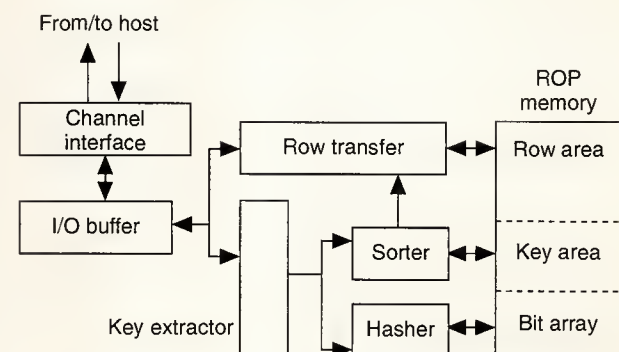


Figure 6. ROP block diagram.

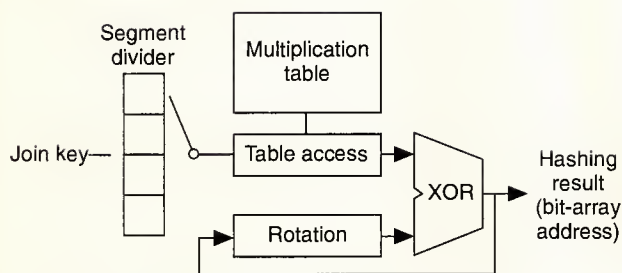


Figure 7. Multiplication-folding function.

method produced good results with fewer collisions for the unknown set of keys. However, in the filtering phase, the division method cannot be applied directly because internal keys may be too long. We determined that Rinda's hashing function should

- operate well when the loading factor, the number of hashing keys over the size of a bit array, is rather small;
- decrease the number of collisions (but it need not handle overflow when a collision occurs); and
- use the same hardware mechanism to hash any keys having various data types and lengths.

For hashing long keys, the folding method works well. It divides keys into several short fragments and folds them using the Exclusive-Or (XOR) operator. However, in character strings, especially in Japanese 16-bit kanji strings, the probability of 1 occurring in each bit is not even. Thus, hashed results have biases if the simple folding method with 1-byte or 2-byte fragments is used. Shifting or reversing the fragments in the bit order solves this problem.

We developed a new multiplication-folding function¹¹ for Rinda: a folding method with bit shifting and multiplication. The multiplication randomizes the character code, and the folding handles long keys. Compact specialized hardware with XOR and simple multiplication circuits implements the method.

Figure 7 is a schematic diagram of the multiplication-folding function. Keys are divided into several short segments. The first segment is multiplied by a prime number and rotated lap-around to distribute the value. Then, the second segment is multiplied by the same prime number and folded on the first segment with XOR. This procedure repeats until all segments are folded. This multiplication-folding function yields a high filtering factor for any

type and length of keys.

Sorting block. Many sorters can increase sorting speed using parallel and pipelined hardware.⁸ However, such hardware sorters are too large for database processors because their hardware size depends on the number of sorted keys. We decided that Rinda's sorter should

- handle a variable number of sorting keys at query execution time,
- achieve a high sorting speed regardless of the number of keys,
- handle any length and type of key efficiently because keys may be long and composed of several data types,
- use compact hardware, and
- permit easy expansion of the allowable number of keys.

The multiway merge sorter¹⁵ using a sorting array¹⁶ meets these requirements. As Figure 8 shows, the sorter implemented in the ROP consists of a sorting array, a merge controller, and working storage, which is the key area of ROP memory. The sorting array is composed of cascade-connected sorting elements with a one-dimensional linear array structure. Each element consists simply of two memories and a comparator as main circuits. The sorting array executes comparison-transferral actions synchronously in a pipelined, parallel manner. Each sorting element compares two keys and selects the smaller or larger one to the neighboring element through a dedicated data path controlled by the previous comparative result.

The sorter can compare a large number of keys in a series of k -way merge operations. In each k -way merging stage, k strings in the working storage are merged at once and then restored as a single string. The size of this string is the total number of keys in all k strings.

Using a new data-driven string-selection technique, the sorter performs a multiway merge operation. Keys, read from the working storage, are identified with bank tags. The sort-

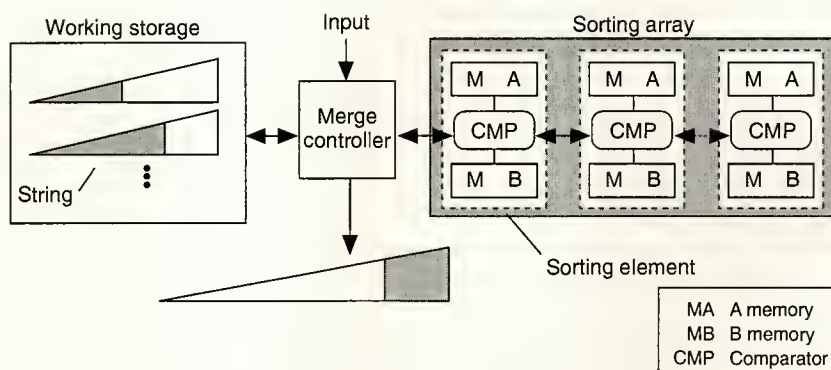


Figure 8. Multiway merge sorter.

Figure 9 shows a continuous key flow using the bank tag. The figure shows an example of four-way merging in descending order. In the M1 phase, the sorting array is filled with the largest keys in all source strings. These largest keys located at the top of the strings. In the M2 phase, merging proceeds successively. The largest key in the sorting array is immediately output with the bank tag. The second-largest candidate key is limited to any of the remaining keys in the sorting array or to the top key in the string indicated by the output key's bank tag. Therefore, this top key is input to the array and is compared with the remaining candidate keys. The sorter performs successive multiway merging using these alternate key output-input operations until all source strings are emptied.

We evaluated Rinda's performance using the extended Wisconsin Benchmark.^{17,18} The database table consisted of 100,000 rows, and each row was 208 bytes long. The Rinda system used the NTT DIPS-V30E superminicomputer as the host, and two CSPs and an ROP. The database resided on two 1.3-Gbyte disks whose data-transfer rate was 3 Mbytes per second.

Results. Figure 10 shows the performance improvement achieved with Rinda. The speedup is the elapsed time without Rinda divided by the elapsed time with Rinda. It is shown with the CPU and I/O times separated. The I/O time includes the processing time of the CSPs and the ROP and the time required to access the work disk. The system executes the queries for 1-percent and 10-percent selections and scalar Min and Count with only the CSPs but performs the others with a combination of the CSPs and the ROP. Table 4 lists the query execution time in seconds with Rinda as well as the

Considerations. Figure 10 shows that the speedup of queries with CSPs depends on the number of result rows. In general, the speedup is smaller as the results are larger. For example, the speedup is more than 50 times in the scalar Count in which no rows are transferred, while it is less than four times in the 10-percent selection. The reason is that the RCP requires more CPU time to transfer rows from the result table to the user program. In other words, the fetch operations take a long time when the number of result rows is large, because the host computer must perform them serially. The speedup of the scalar Min is less than that of the scalar Count, because the Count is directly supported by the CSPs while the Min is not. As for the Min, the specified column's values of all rows are transferred from the CSPs to

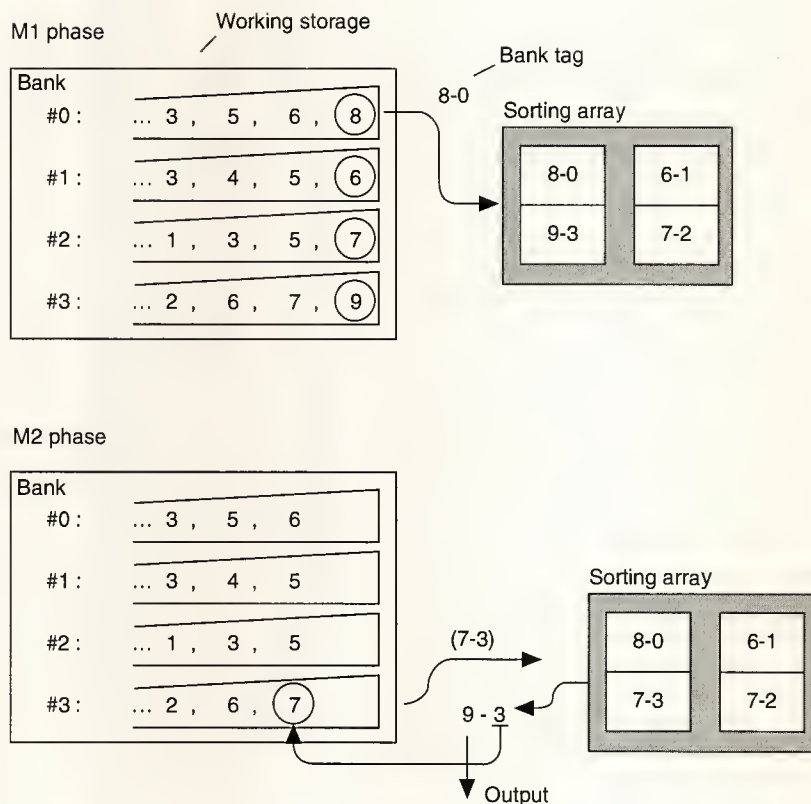


Figure 9. Continuous multiway merging using bank tags.

Table 3. SQL statements executed.

Query	Type	SQL statement
Selection	1 percent	SELECT * FROM HUNKA WHERE UNIQUE1>=50000 AND UNIQUE1<51000
	10 percent	SELECT * FROM HUNKA WHERE UNIQUE1>=50000 AND UNIQUE1<60000
Join	AselB	SELECT * FROM HUNKA A, HUNKB B WHERE A.UNIQUE1=B.UNIQUE1 AND A.UNIQUE1<10000
	CselAselB	SELECT * FROM HUNKA A, HUNKB B, TENKA C WHERE C.UNIQUE1=A.UNIQUE1 AND A.UNIQUE1=B.UNIQUE1 AND A.UNIQUE1<10000 AND B.UNIQUE1<10000
Min	Scalar	SELECT MIN(UNIQUE1) FROM HUNKA
	Group-By	SELECT MIN(TWOTHOU5) FROM HUNKA GROUP BY HUNDRED
Count	Scalar	SELECT COUNT(*) FROM HUNKA
	Group-By	SELECT COUNT(*) FROM HUNKA GROUP BY HUNDRED

the RCP, which determines the minimum value.

Comparison of the scalar and Group-By in the Min or Count functions in Figure 10 shows the speedup achieved with an ROP. The additional sorting load does not reduce the speedup, because the ROP sorts dozens of times faster than the host computer. The join queries also demonstrate the ROP's effect in contrast with the 10-percent selection, which does not use the ROP.

Table 4 shows that Rinda is faster in the selections and slower in the joins and Min functions than the Teradata and Gamma machines. The times differ because the host computer has almost nothing to do for the selections, while it must perform the merge-join phase for the joins and value comparisons for the Min function. These loads are somewhat smaller than those of the original queries. However, they still require relatively long processing times in single small-scale general-purpose processors.

Table 4. Query execution times in seconds.

Query	Type	Rinda	Teradata	Gamma
Selection	1 percent	5.2	18.6	13.4
	10 percent	9.0	14.9	12.7
Join	AselB	136.8	235.6	35.8
	CselAselB	128.5	95.7	37.9
MIN	Scalar	31.7	18.3	15.5
	Group-By	47.3	27.1	19.4

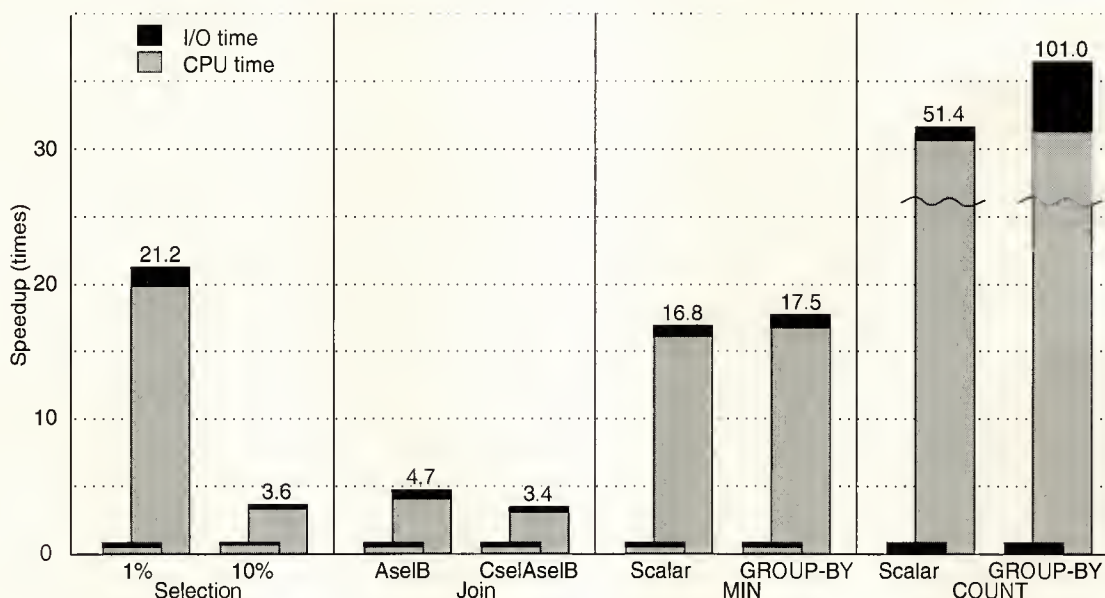


Figure 10. Speedup with Rinda (left) and without Rinda (right).

Merits and limitations

Rinda accelerates searching and sorting 20 to 50 times, and its total speedup of queries is in the range of three to 100 times. Rinda is generally very efficient when the source table is large and the result table is small.

However, Rinda has limitations, the first of which is concurrent execution with updates. To guarantee transaction serializability, the RCP (Rinda control program) must lock the whole table before CSP searching, resulting in a locking overhead with nonindexed queries and time delays with indexed updates. Using a dirty read facility that requires no locks for queries executed by Rinda solves this problem. The second limitation involves multiple query execution. The RCP executes multiple queries concurrently, while each CSP and ROP accepts only one request at a time. Therefore, simultaneous execution of two queries that require the same CSP or ROP increases response time.

RINDA RELIEVES ITS HOST COMPUTER OF THE HEAVY loads caused by nonindexed queries. The CSP, a special-purpose processor with hardware for searching, selects rows from a disk at the disk's data-transfer rate. The ROP, another processor with specialized hardware, sorts rows selected by the CSPs. For join queries, the ROP uses the three-phase join method with a hardware filter and sorter. Rinda substantially reduces the host computer's CPU and I/O times. Our performance study shows that Rinda accelerates nonindexed queries from three to 100 times, compared with conventional DBMS software.

Rinda operates with several business database systems. The businesses use Rinda mainly to retrieve rows by ad hoc queries or to get statistical reports from very large tables. Rinda's role is increasing as database applications become more advanced. ■

Acknowledgments

We thank Masato Haihara and Kenji Suzuki of NTT for their continuous support of this work.

References

1. E. Babb, "Implementing a Relational Database by Means of Specialized Hardware," *ACM Trans. Database Systems*, Vol. 4, No. 1, Mar. 1979, pp. 1-29.
2. K. Kojima, S. Torii, and S. Yoshizumi, "IDP—A Main Storage Based Vector Database Processor," *Database Machines and Knowledge Base Machines*, M. Kitsuregawa and H. Tanaka, eds., Kluwer Academic, Boston, 1988, pp. 47-60.
3. U. Inoue et al., "RINDA—A Relational Database Processor for Non-Indexed Queries," *Proc. First Int'l Symp. Database Systems for Advanced Applications*, Apr. 1989, pp. 382-386.
4. U. Inoue and S. Kawazu, "A Relational Database Machine for Very Large Information Retrieval Systems," in *Database Machines*, NATO ASI Series, Vol. F24, A.K. Sood and A.H. Qureshi, eds., Springer-Verlag, Berlin, 1986, pp. 183-201.
5. H. Takeda and T. Satoh, "An Accelerating Processor for Relational Operations," *Proc. Int'l Conf. Databases, Parallel Architectures, and Their Applications*, Mar. 1990, p. 559.
6. S. Shiokawa, Y. Obashi, and A. Nagoya, "DIPS-11/5E Series Mainframe," *Review of the ECL*, Vol. 35, No. 6, June 1987, pp. 633-641.
7. K. Itoh, R. Yazawa, and Y. Fukumura, "DIPS-V30 Development," *Review of the ECL*, Vol. 34, No. 5, May 1986, pp. 587-593.
8. S.G. Akl, *Parallel Sorting Algorithms*, Academic Press, New York, 1985.
9. M.W. Blasgen and K.P. Eswaran, "Storage and Access in Relational Data Bases," *IBM System J.*, Vol. 16, No. 4, 1977, pp. 363-377.
10. M. Kitsuregawa, M. Tanaka, and T. Moto-oka, "Application of Hash to Data Base Machine and Its Architecture," *New Generation Computing*, Vol. 1, No. 1, 1983, pp. 63-74.
11. T. Satoh et al., "Acceleration of Join Operations by a Relational Database Processor, RINDA," *Proc. Second Int'l Symp. Database Systems for Advanced Applications*, Apr. 1991, pp. 243-248.
12. D.R. McGregor, R.G. Thomson, and W.N. Dawson, "High Performance Hardware for Database Systems," in *Systems for Large Data Bases*, P.C. Lockemann and E.J. Neuhold, eds., North-Holland, Amsterdam, 1976, pp. 103-116.
13. V.Y. Lum, P.S.T. Yuen, and M. Dodd, "Key-to-Address Transform Techniques: A Fundamental Performance Study on Large Existing Formatted Files," *Comm. ACM*, Vol. 14, No. 4, Apr. 1971, pp. 228-239.
14. G.D. Knott, "Hashing Functions," *Computer J.*, Vol. 18, No. 3, Aug. 1975, pp. 265-287.
15. T. Satoh, H. Takeda, and N. Tsuda, "A Compact Multiway Merge Sorter Using VLSI Linear-Array Comparators," *Proc. Int'l Conf. Foundation Data Organization and Algorithms*, June 1989, pp. 223-227.
16. N. Tsuda, T. Satoh, and T. Kawada, "A Pipeline Sorting Chip," *Proc. IEEE Int'l Solid-State Circuits Conf.*, IEEE CS Press, Los Alamitos, Calif., 1987, pp. 270-271.
17. D. Bitton, D.J. DeWitt, and C. Turbyfill, "Benchmarking Database Systems—A Systematic Approach," CTSR #526, Univ. of Wisconsin-

Madison, 1983.

18. D.J. DeWitt et al., "A Single User Evaluation of the GAMMA Database Machine," in *Database Machines and Knowledge Base Machines*, M. Kitsuregawa and H. Tanaka, eds., Kluwer Academic, 1988, pp. 370-386.



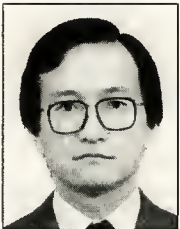
Ushio Inoue is a senior research engineer at NTT Communications and Information Processing Laboratories. His research interests include information retrieval systems, database-management systems, real-time database systems, and database machines.

Inoue received a BE in electrical engineering from Nagoya University. He is a member of the IEEE Computer Society, the Institute of Electronics, Information and Communication Engineers of Japan, and the Information Processing Society of Japan.



Tetsuji Satoh is a senior research engineer at NTT Communications and Information Processing Laboratories. His research interests include wafer-scale integration, parallel-processing architectures, hardware sorters, and database machines.

Satoh received a BE in electronic engineering from Yamanashi University. He is a member of the IEEE Computer Society, the Institute of Electronics, Information and Communication Engineers of Japan, and the Information Processing Society of Japan.



Haruo Hayami is a senior research engineer at NTT Communications and Information Processing Laboratories. His research interests include mainframes and database machines.

Hayami received a BS and an MS in applied physics from Nagoya University. He is a member of the Information Processing Society of Japan.



Hideaki Takeda is a senior research engineer at NTT Communications and Information Processing Laboratories. His research interests include parallel database algorithms, hardware sorters, and database machines.

Takeda received a BE and an ME in electrical engineering from Hokkaido University. He is a member of the IEEE Computer Society and the Information Processing Society of Japan.



Toshio Nakamura is a senior research engineer at NTT Communications and Information Processing Laboratories. His research interests include compilers, database-management systems, and database machines.

Nakamura received a BE in electrical engineering from the Kyoto Institute of Technology. He is a member of the Institute of Electronics, Information and Communication Engineers of Japan and the Information Processing Society of Japan.



Hideki Fukuoka is a senior research engineer at NTT Communications and Information Processing Laboratories. His research interests include mainframes and database machines.

Fukuoka received a BE in electronic engineering from the University of Osaka Prefecture. He is a member of the Institute of Electronics, Information and Communication Engineers of Japan and the Information Processing Society of Japan.

Address questions regarding this article to Ushio Inoue, NTT Communications and Information Processing Laboratories, 1-2356 Take, Yokosuka-shi, Kanagawa 238-03, Japan; inoue@syrinx.ntt.jp.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate numbers on the Reader Service Card.

Low 165

Medium 166

High 167

1991 Index

IEEE Micro



This index covers all technical items — papers, correspondence, reviews, etc. — that appeared in *IEEE Micro* during 1991, and items from previous years that were commented upon or corrected in 1991.

The *Author Index* contains the primary entry for each item, listed under the first author's name, and cross-references from all coauthors. The *Subject Index* contains several entries for each item under appropriate subject headings, and subject cross-references.

It is always necessary to refer to the primary entry in the *Author Index* for the exact title, coauthors, and comments/corrections.

AUTHOR INDEX

A

- Abdelguerfi, M.**, and A. K. Sood. A fine grain architecture for relational database aggregation operations; *M-M Dec 91* 35-43
- Aschmann, Hans-Ruedi**, Niklaus Giger, Elisabeth Hoepli, Peter Janak, and Hubert Kirmann. Alphorn: A remote procedure call environment for fault-tolerant, heterogeneous, distributed systems; *M-M Oct 91* 16-19, 60-67
- Atkins, Mark**. Performance and the i860 microprocessor; *M-M Oct 91* 24-27, 72-78

B

- Balestra, Gabriella**, see Knaflitz, Marco, *M-M Oct 91* 12-15, 48-58
- Blasgen, Michael W.**, see Oehler, Richard R., *M-M Jun 91* 14-17, 56-62
- Brookwood, Nathan A.**, see Sachs, Howard G., *M-M Jun 91* 18-21, 74-80

C

- Caesar, Knut**, see Schmidt, Ulrich, *M-M Jun 91* 22-25, 88-94
- Chen, Ching-Ho**, see Li, Hua, *M-M Oct 91* 8-11, 44-47
- Chia, Wei-Kuo**, see Kuo, Yau-Hwang, *M-M Aug 91* 32-35, 86-92

D

- Di Stefano, Antonella**, Orazio Mirabella, and Fabio Presente. Implementing a DSP-based Petri-net simulation tool; *M-M Apr 91* 20-23, 56-64

F

- Farrell, James J., III**, see Seaborn, True, *M-M Feb 91* 5-9, 61

Faudemay, Pascal, and Mongia Mhiri. An associative accelerator for large data bases; *M-M Dec 91* 22-34

Fukuda, Akira, Kazuaki Murakami, and Shinji Tomita. Toward advanced parallel processing: Exploiting parallelism at task and instruction levels; *M-M Aug 91* 16-19, 50-61

Fukuoka, Hideki, see Inoue, Ushio, *M-M Dec 91* 61-70

Fulcher, John A. Fun and games and microcomputer interfacing; *M-M Feb 91* 18-21, 75-78

G

Gibson, Gary, see Popescu, Val, *M-M Jun 91* 10-13, 63-73

Giger, Niklaus, see Aschmann, Hans-Ruedi, *M-M Oct 91* 16-19, 60-67

H

- Haden, Kirtley B.**, see Russell, David W., *M-M Feb 91* 26-29
- Hall, Douglas V.** Adapting curriculum materials for different course sequences; *M-M Feb 91* 34-37, 82-83
- Hanson, Lee F.**, see Sachs, Howard G., *M-M Jun 91* 18-21, 74-80
- Hardie, Mark S.**, see Peyton Jones, Simon L., *M-M Feb 91* 38-41, 84-93
- Hayami, Haruo**, see Inoue, Ushio, *M-M Dec 91* 61-70
- Herman, Gary E.**, see Lee, Kuo Chu, *M-M Dec 91* 8-20
- Hickey, Takako Matoba**, see Lee, Kuo Chu, *M-M Dec 91* 8-20
- Hill, Mark D.**, and David A. Wood. Guest Editors' introduction: Hot chips II symposium (Special issue intro.); *M-M Jun 91* 8-9
- Hinata, Junichi**, see Yoshida, Toyohiko, *M-M Aug 91* 20-23, 62-72
- Hoepli, Elisabeth**, see Aschmann, Hans-Ruedi, *M-M Oct 91* 16-19, 60-67
- Hootman, Joe**, see Seaborn, True, *M-M Feb 91* 5-9, 61
- Hsiao, David K.** A parallel, scalable, and microprocessor-based database computer for performance gains and capacity growth; *M-M Dec 91* 44-60

I

- Inoue, Ushio**, Tetsuji Satoh, Haruo Hayami, Hideaki Takeda, Toshio Nakamura, and Hideki Fukuoka. RINDA: A relational database processor with hardware specialized for searching and sorting; *M-M Dec 91* 61-70

Isaman, David, see Popescu, Val, *M-M Jun 91* 10-13, 63-73

J

- Jaeger, Richard C.**, see Seaborn, True, *M-M Feb 91* 5-9, 61
- Janak, Peter**, see Aschmann, Hans-Ruedi, *M-M Oct 91* 16-19, 60-70
- Jeng, Guang-Huei**, see Kuo, Yau-Hwang, *M-M Aug 91* 32-35, 86-92

Jordan, Ramiro, *see* Pollard, L. Howard, *M-M Feb 91* 22-25, 78-79

K

- Kabemoto, Akira**, and Hiroshi Yoshida. The architecture of the Sure System 2000 communications processor; *M-M Aug 91* 28-31, 73-78
- Kahaner, David K.** Software Report—Optical computing activities; *M-M Feb 91* 53-56
- Kahaner, David K.** Software Report—Computer growth in Taiwan; *M-M Jun 91* 39-41
- Kahaner, David K.** Software Report—A glimpse of the future; *M-M Oct 91* 35, 79
- Kirrmann, Hubert.** Micro World—When the computer was still personal; *M-M Feb 91* 42-44
- Kirrmann, Hubert.** Micro World—Light at the end of the chunnel; *M-M Jun 91* 4-6
- Kirrmann, Hubert.** Micro World—Europe on the way to the metric system; *M-M Aug 91* 36-39
- Kirrmann, Hubert.** *see* Aschmann, Hans-Ruedi, *M-M Oct 91* 16-19, 60-67
- Klir, George J.** Software Report—Advances in fuzzy theory and applications; *M-M Aug 91* 8-11
- Knaflitz, Marco**, and Gabriella Balestra. Computer analysis of the myoelectric signal; *M-M Oct 91* 12-15, 48-58
- Kung, Ling-Yang.** *see* Kuo, Yau-Hwang, *M-M Aug 91* 32-35, 86-92
- Kuo, Yau-Hwang**, Ling-Yang Kung, Ching-Chung Tzeng, Guang-Huei Jeng, and Wei-Kuo Chia. KMDS: An expert system for integrated hardware/software design of microprocessor-based digital systems; *M-M Aug 91* 32-35, 86-92

L

- Lee, Kuo Chu**, Takako Matoba Hickey, Victor W. Mak, and Gary E. Herman. VLSI accelerators for large database systems; *M-M Dec 91* 8-20
- Lehoczky, John P.**, *see* Sha, Lui, *M-M Jun 91* 30-33, 95-100
- Li, Hua**, and Ching-Ho Chen. Simulating a function of visual peripheral processes with an analog VLSI network; *M-M Oct 91* 8-11, 44-47
- Lightner, Bruce.** *see* Popescu, Val, *M-M Jun 91* 10-13, 63-73
- Louri, Ahmed.** Three-dimensional optical architecture and data-parallel algorithms for massively parallel computing; *M-M Apr 91* 24-27, 65-82

M

- Mak, Victor W.**, *see* Lee, Kuo Chu, *M-M Dec 91* 8-20
- Mateosian, Richard.** Review of 'Postscript Language Reference Manual, 2nd edn.' (Systems, A.; 1990); *M-M Apr 91* 28-29
- Mateosian, Richard.** Review of 'Intel's Official Guide to 386 Computing' (Edelhart, M.; 1991); *M-M Jun 91* 50-51
- Mateosian, Richard.** Review of 'Superscalar Microprocessor Design' (Johnson, M.); *M-M Jun 91* 51, 102
- Mateosian, Richard.** Review of 'The Emperor's New Mind' (Penrose, R.; 1991); *M-M Aug 91* 42-43
- Mateosian, Richard.** Review of 'Computer Dictionary'; *M-M Aug 91* 43
- Mateosian, Richard.** Review of 'The Creative Mind—Myths and Mechanisms' (Boden, M. A.; 1990); *M-M Oct 91* 4-6
- McGhan, Harlan.** *see* Sachs, Howard G., *M-M Jun 91* 18-21, 74-80
- Mhiri, Mongia.** *see* Faudemay, Pascal, *M-M Dec 91* 22-34
- Mirabella, Orazio.** *see* Di Stefano, Antonella, *M-M Apr 91* 20-23, 56-64
- Mizugaki, Shigeo.** *see* Yoshida, Toyohiko, *M-M Aug 91* 20-23, 62-72
- Murakami, Kazuaki.** *see* Fukuda, Akira, *M-M Aug 91* 16-19, 50-61
- Myers, Ware.** The drive to the year 2000; *M-M Feb 91* 10-13, 68-74

N

- Nakamura, Toshio.** *see* Inoue, Ushio, *M-M Dec 91* 61-70
- Nicoud, Jean-Daniel.** Dedicated tools for microprocessor education; *M-M Feb 91* 14-17, 62-68

O

- Oehler, Richard R.**, and Michael W. Blasgen. IBM RISC System/6000: Architecture and performance; *M-M Jun 91* 14-17, 56-62

P

- Peterson, Craig**, James Sutton, and Paul Wiley. iWarp: A 100-MOPS, LIW microprocessor for multicomputers; *M-M Jun 91* 26-29, 81-87
- Peyton Jones, Simon L.**, and Mark S. Hardie. A Futurebus interface from off-the-shelf parts; *M-M Feb 91* 38-41, 84-93
- Pollard, L. Howard**, and Ramiro Jordan. An advanced educational microprocessor system; *M-M Feb 91* 22-25, 78-79
- Popescu, Val**, Merle Schultz, John Spracklen, Gary Gibson, Bruce Lightner, and David Isaman. The metaflow architecture; *M-M Jun 91* 10-13, 63-73
- Presente, Fabio.** *see* Di Stefano, Antonella, *M-M Apr 91* 20-23, 56-64
- Prete, Cosimo A.** RST cache memory design for a tightly coupled multiprocessor system; *M-M Apr 91* 16-19, 40-52

R

- Rajkumar, Ragunathan.** *see* Sha, Lui, *M-M Jun 91* 30-33, 95-100
- Roberts, Charles E.** A RISC processor for embedded applications within an ASIC; *M-M Oct 91* 20-23, 68-72
- Rony, Peter R.**, *see* Seaborn, True, *M-M Feb 91* 5-9, 61
- Russell, David W.**, and Kirtley B. Haden. A configurable, virtual microprocessor system for instructional use in real-time, real-world studies; *M-M Feb 91* 26-29

S

- Sachs, Howard G.**, Harlan McGhan, Lee F. Hanson, and Nathan A. Brookwood. Design and implementation trade-offs in the Clipper C400 architecture; *M-M Jun 91* 18-21, 74-80
- Sakamura, Ken.** Guest editor's introduction: Presenting the Far East special issue for 1991; *M-M Aug 91* 12-15
- Sakamura, Ken.** *see* Takada, Hiroaki, *M-M Aug 91* 24-27, 78-85
- Satoh, Tetsuji.** *see* Inoue, Ushio, *M-M Dec 91* 61-70
- Schmidt, Ulrich**, and Knut Caesar. Datawave: A single-chip multiprocessor for video applications; *M-M Jun 91* 22-25, 88-94
- Schultz, Merle.** *see* Popescu, Val, *M-M Jun 91* 10-13, 63-73
- Schultz, Thomas W.** Peripheral hardware and a hands-on multitasking lab; *M-M Feb 91* 30-33, 80-82
- Seaborn, True**, Peter R. Rony, Richard C. Jaeger, James J. Farrell III, and Joe Hootman. Looking back; *M-M Feb 91* 5-9, 61
- Sha, Lui**, Ragunathan Rajkumar, and John P. Lehoczky. Real-time computing with IEEE Futurebus+; *M-M Jun 91* 30-33, 95-100
- Shimizu, Toru.** *see* Yoshida, Toyohiko, *M-M Aug 91* 20-23, 62-72
- Slater, Michael.** Micro View—The end of the 386 monopoly; *M-M Apr 91* 88, 87
- Slater, Michael.** Micro View—Evolving architectures; *M-M Feb 91* 96-95
- Sood, A. K.**, *see* Abdelguerfi, M., *M-M Dec 91* 35-43
- Spracklen, John.** *see* Popescu, Val, *M-M Jun 91* 10-13, 63-73
- Stern, Richard H.** Micro Law—The paperback case; *M-M Apr 91* 30-33
- Stern, Richard H.** Micro Law—(C):Software\Legal.hlp!; *M-M Jun 91* 42-46
- Stern, Richard H.** Micro Law—The first chip-layout copying case; *M-M Aug 91* 3-6, 94
- Stern, Richard H.** Micro Law—Fraud on the copyright office; *M-M Oct 91* 33-34
- Stern, Richard H.** Micro Law—Database system copyrights; *M-M Dec 91* 77-79
- Stern, Richard M.** Micro Law—The paperback case; *M-M Feb 91* 48-51
- Sutton, James.** *see* Peterson, Craig, *M-M Jun 91* 26-29, 81-87

T

- Takada, Hiroaki**, and Ken Sakamura. ITRON-MP: An adaptive real-time kernel specification for shared-memory multiprocessor systems; *M-M Aug 91* 24-27, 78-85
Takeda, Hideaki, see Inoue, Ushio, *M-M Dec 91* 61-70
Tomita, Shinji, see Fukuda, Akira, *M-M Aug 91* 16-19, 50-61
Tzeng, Ching-Chung, see Kuo, Yau-Hwang, *M-M Aug 91* 32-35, 86-92

W

- Warren, Carl**. Micro Standards—There's a standard hiding out there; *M-M Feb 91* 45, 56
Warren, Carl. On the Edge—Evaluating shielded twisted-pair cable; *M-M Feb 91* 46-47
Warren, Carl. Micro Standards—Sorry, Captain Kirk; *M-M Apr 91* 34-35
Warren, Carl. On the Edge—Rate monotonic scheduling; *M-M Jun 91* 34-38, 102
Warren, Carl. Micro Standards—A bountiful return; *M-M Aug 91* 40-41
Warren, Carl. On the Edge—IEEE standard P1754: An open microprocessor architecture; *M-M Oct 91* 30-33
Warren, Carl. Micro Standards—Computing interconnections; *M-M Dec 91* 83-85
Wiley, Paul, see Peterson, Craig, *M-M Jun 91* 26-29, 81-87
Wood, David A., see Hill, Mark D., *M-M Jun 91* 8-9

Y

- Yoshida, Hiroshi**, see Kabemoto, Akira, *M-M Aug 91* 28-31, 73-78
Yoshida, Toyohiko, Toru Shimizu, Shigeo Mizugaki, and Junichi Hinata. The Gmicro/100 32-bit microprocessor; *M-M Aug 91* 20-23, 62-72

Z

- Zhang, Xiaodong**. System effects of interprocessor communication latency in multicomputers; *M-M Apr 91* 12-15, 52-55. *Correction*, *Jun 91* 6
Zhang, Xiaodong. Correction to 'System Effects of Interprocessor Communication Latency in Multicomputers' (Apr 91 12-15, 52-55); *M-M Jun 91* 6

SUBJECT INDEX

A

- Analog computers; cf.** Optical computing
Animation; cf. Visualization
Application-specific integrated circuits
 RISC processor for embedded applications within ASIC. *Roberts, Charles E.*, *M-M Oct 91* 20-23, 68-72
Array processing
 Datawave single-chip multiprocessor for video applications. *Schmidt, Ulrich*, +, *M-M Jun 91* 22-25, 88-94
Artificial intelligence; cf. Cognitive science
Associative memories
 associative caches and associative circuits as accelerators for large databases. *Faudemay, Pascal*, +, *M-M Dec 91* 22-34

B

- Bibliographies**
 3-D optical architecture. *Louri, Ahmed*, *M-M Apr 91* 24-27, 65-82
Biomedical signal analysis
 myoelectric signal analysis using computer systems and signal processing techniques. *Knaflitz, Marco*, +, *M-M Oct 91* 12-15, 48-58
Book reviews
 Computer Dictionary. *Mateosian, Richard*, *M-M Aug 91* 43

- Intel's Official Guide to 386 Computing (Edelhart, M.; 1991). *Mateosian, Richard*, *M-M Jun 91* 50-51
 Postscript Language Reference Manual, 2nd edn. (Systems, A.; 1990). *Mateosian, Richard*, *M-M Apr 91* 28-29
 Superscalar Microprocessor Design (Johnson, M.). *Mateosian, Richard*, *M-M Jun 91* 51, 102
 The Creative Mind—Myths and Mechanisms (Boden, M. A.; 1990). *Mateosian, Richard*, *M-M Oct 91* 4-6
 The Emperor's New Mind (Penrose, R.; 1991). *Mateosian, Richard*, *M-M Aug 91* 42-43

- Brain; cf.** Cognitive science
Buffer memories; cf. Cache memories
Bulk memories; cf. Mass memories
Business; cf. Computer industry

C

- Cable shielding; cf.** Wire communication cable shielding
Cache memories
 associative caches and associative circuits as accelerators for large databases. *Faudemay, Pascal*, +, *M-M Dec 91* 22-34
 RST cache memory design for tightly coupled Clipper-based multiprocessor system. *Prete, Cosimo A.*, *M-M Apr 91* 16-19, 40-52
Cellular logic
 Datawave single-chip multiprocessor for video applications. *Schmidt, Ulrich*, +, *M-M Jun 91* 22-25, 88-94
Circuit simulation
 KMDS expert system for integrated hardware/software design of microprocessor-based digital systems. *Kuo, Yau-Hwang*, +, *M-M Aug 91* 32-35, 86-92
Cognitive science
 book review: The Creative Mind—Myths and Mechanisms (Boden, M. A.; 1990). *Mateosian, Richard*, *M-M Oct 91* 4-6
 book review: The Emperor's New Mind (Penrose, R.; 1991). *Mateosian, Richard*, *M-M Aug 91* 42-43
Communication switching; cf. Message switching
Communication system performance; cf. Computer network performance
Compilers
 Metaflow architecture using deferred scheduling and register-renaming instruction shelf to manage out-of-order execution. *Popescu, Val*, +, *M-M Jun 91* 10-13, 63-73
Computer applications
 project management, site, track, and mobile computers used in construction of England-to-France tunnel (Micro World). *Kirmann, Hubert*, *M-M Jun 91* 4-6
Computer architecture
 3-D optical architecture. *Louri, Ahmed*, *M-M Apr 91* 24-27, 65-82
 evolving architectures for microprocessors; overview. *Slater, Michael*, *M-M Feb 91* 96-95
 IBM RISC System/6000 architecture and performance; benchmark details. *Oehler, Richard R.*, +, *M-M Jun 91* 14-17, 56-62
 KRPP and DSNS superscalar architectures using task and instruction level parallelism. *Fukuda, Akira*, +, *M-M Aug 91* 16-19, 50-61
 Metaflow architecture using deferred scheduling and register-renaming instruction shelf to manage out-of-order execution. *Popescu, Val*, +, *M-M Jun 91* 10-13, 63-73
 redesign of Clipper C400 RISC architecture. *Sachs, Howard G.*, +, *M-M Jun 91* 18-21, 74-80
Computer buses; cf. Data buses
Computer control; cf. Digital control
Computer engineering education; cf. Computer science education
Computer graphics; cf. Visual languages; Visualization
Computer graphics languages; cf. Visual languages
Computer industry
 growth of computer industry in Taiwan (Software Report). *Kahaner, David K.*, *M-M Jun 91* 39-41

Computer interfaces

IEEE P1175/D11 draft standard on computing system tool interconnections overview (Micro Standards). *Warren, Carl, M-M Dec 91 83-85*

Lotus Development Corp. vs. Paperback Software International; court's failure to recognize functionality of standardization (Micro Law). *Stern, Richard M., M-M Feb 91 48-51*

real-time computing with IEEE Futurebus+. *Sha, Lui, +, M-M Jun 91 30-33, 95-100*

Computer interfaces; cf. Measurement-system data handling; Microcomputer interfaces

Computer language processors; cf. Compilers

Computer languages

Alphorn remote procedure call environment for fault-tolerant, heterogeneous, distributed systems. *Aschmann, Hans-Ruedi, +, M-M Oct 91 16-19, 60-67*

book review; Postscript Language Reference Manual, 2nd edn. (Systems, A.; 1990). *Mateosian, Richard, M-M Apr 91 28-29*

Computer languages; cf. Visual languages

Computer network performance

interprocessor communication performance evaluation of five types of hypercube and grid-topology multicomputers. *Zhang, Xiaodong, M-M Apr 91 12-15, 52-55+*

Computer networks; cf. Local area networks; Multiprocessing

Computer peripherals; cf. Microcomputer peripherals

Computer pipeline processing; cf. Pipeline processing

Computer science

book review; Computer Dictionary. *Mateosian, Richard, M-M Aug 91 43*

Computer science education

adapting curriculum materials for different sequences of microprocessing courses. *Hall, Douglas V., M-M Feb 91 34-37, 82-83*

advanced educational microprocessor system used as classroom demonstration tool and laboratory instrument. *Pollard, L. Howard, +, M-M Feb 91 22-25, 78-79*

microcomputer-interfacing laboratory projects. *Fulcher, John A., M-M Feb 91 18-21, 75-78*

microprocessor education curriculum incorporating logidules, CALM language, and Dauphin and Smaky microprocessor systems. *Nicoud, Jean-Daniel, M-M Feb 91 14-17, 62-68*

microprocessors in education (special issue). *M-M Feb 91 14-83*

teaching methods for peripheral hardware class and hands-on multitasking lab. *Schultz, Thomas W., M-M Feb 91 30-33, 80-82*

Computer vision; cf. Machine vision

Computers; cf. Database machines; Distributed computing; Microcomputers; Optical computing; Parallel processing; Virtual computers

Content-addressable memories; cf. Associative memories

Control engineering education

configurable, virtual microprocessor system to simulate and validate process plant designs in classroom. *Russell, David W., +, M-M Feb 91 26-29*

Control systems; cf. Digital control

Copyright protection

answering questions on copyright office forms correctly (Micro Law). *Stern, Richard H., M-M Oct 91 33-34*

database system copyrights (Micro Law). *Stern, Richard H., M-M Dec 91 77-79*

Copyright protection; cf. Software protection

Custom integrated circuits; cf. Application-specific integrated circuits

D

Data acquisition; cf. Measurement-system data handling

Data buses

IEEE P996 draft standard for AT-bus; development (Micro Standards). *Warren, Carl, M-M Feb 91 45, 56*

Data communication; cf. Distributed computing; Local area networks; Measurement-system data handling; Message switching; Multiprocessing, interconnection

Data processing; cf. Database systems

Data structures

associative caches and associative circuits as accelerators for large databases. *Faudemay, Pascal, +, M-M Dec 91 22-34*

Database machines

associative caches and associative circuits as accelerators for large databases. *Faudemay, Pascal, +, M-M Dec 91 22-34*

database machines (special issue). *M-M Dec 91 6-76*

multibackend database supercomputer interconnected by LAN; architecture and performance. *Hsiao, David K., M-M Dec 91 44-60*

RINDA relational database processor with hardware specialized for searching and sorting. *Inoue, Ushio, +, M-M Dec 91 61-70*

Database management systems; cf. Database systems, searching

Database systems

database system copyrights (Micro Law). *Stern, Richard H., M-M Dec 91 77-79*

Database systems; cf. Distributed database systems

Database systems, relational

fine grain architecture for relational database aggregation. *Abdelguerfi, M., +, M-M Dec 91 35-43*

RINDA relational database processor with hardware specialized for searching and sorting. *Inoue, Ushio, +, M-M Dec 91 61-70*

VLSI accelerators for improving large database system performance. *Lee, Kuo Chu, +, M-M Dec 91 8-20*

Database systems, searching

RINDA relational database processor with hardware specialized for searching and sorting. *Inoue, Ushio, +, M-M Dec 91 61-70*

VLSI accelerators for improving large database system performance. *Lee, Kuo Chu, +, M-M Dec 91 8-20*

Design automation; cf. Circuit simulation; Design automation software

Design automation software

shielded twisted-pair cable characteristics evaluation using Math CAD (On the Edge). *Warren, Carl, M-M Feb 91 46-47*

Digital control

configurable, virtual microprocessor system to simulate and validate process plant designs in classroom. *Russell, David W., +, M-M Feb 91 26-29*

Digital integrated circuits; cf. Very-large-scale integration

Digital signal processors; cf. Microprocessors

Disk recording; cf. Optical memories

Distributed computing

iWarp microprocessor supporting message-passing and systolic communications for multicomputers. *Peterson, Craig, +, M-M Jun 91 26-29, 81-87*

Distributed computing; cf. Distributed database systems; Multiprocessing

Distributed database systems

Alphorn remote procedure call environment for fault-tolerant, heterogeneous, distributed systems. *Aschmann, Hans-Ruedi, +, M-M Oct 91 16-19, 60-67*

multibackend database supercomputer interconnected by LAN; architecture and performance. *Hsiao, David K., M-M Dec 91 44-60*

E**Education**

microprocessors in education (special issue). *M-M Feb 91 14-83*

Education; cf. Computer science education

Educational technology; cf. Computer science education; Visualization

Electrical engineering education; cf. Control engineering education

EMG (electromyography); cf. Muscles, EMG

England; cf. United Kingdom

Europe

use of English units in European hardware and software systems (Micro World). *Kirrmann, Hubert, M-M Aug 91 36-39*

Expert systems

KMDS expert system for integrated hardware/software design of microprocessor-based digital systems. *Kuo, Yau-Hwang, + , M-M Aug 91 32-35, 86-92*

F

Fault tolerance; cf. Redundant systems

Forecasting; cf. Technology forecasting

France

project management, site, track, and mobile computers used in construction of England-to-France tunnel (Micro World). *Kirrmann, Hubert, M-M Jun 91 4-6*

Fuzzy set theory

fuzzy set theory applications and advances in Japan (Software Report). *Klir, George J., M-M Aug 91 8-11*

H

History

microcomputer developments during past ten years (Micro World). *Kirrmann, Hubert, M-M Feb 91 42-44*

tenth anniversary retrospective. *Seaborn, True, + , M-M Feb 91 5-9, 61*

Holographic memories

3-D optical architecture. *Louri, Ahmed, M-M Apr 91 24-27, 65-82*

Human factors; cf. Cognitive science

I

IEEE Micro

tenth anniversary retrospective. *Seaborn, True, + , M-M Feb 91 5-9, 61*

IEEE standards

developments in several draft standards (Micro Standards). *Warren, Carl, M-M Aug 91 40-41*

IEEE open microprocessor architecture standard P1754 (On the Edge). *Warren, Carl, M-M Oct 91 30-33*

IEEE P1175/D11 draft standard on computing system tool interconnections overview (Micro Standards). *Warren, Carl, M-M Dec 91 83-85*

IEEE P996 draft standard for AT-bus; development (Micro Standards). *Warren, Carl, M-M Feb 91 45, 56*

real-time computing with IEEE Futurebus+. *Sha, Lui, + , M-M Jun 91 30-33, 95-100*

Information systems; cf. Database systems

Instrumentation; cf. Measurement

Integrated circuits; cf. Application-specific integrated circuits; Microprocessors; Very-large-scale integration

J

Japan

Far East (special issue). *M-M Aug 91 12-31*

fuzzy set theory applications and advances in Japan (Software Report). *Klir, George J., M-M Aug 91 8-11*

K

Knowledge-based systems; cf. Expert systems

L

LAN; cf. Local area networks

Languages; cf. Computer languages

Learning systems; cf. Neural networks

Legal factors

Brooktree Corp. vs. Advanced Micro Devices, Inc.; issues arising from first chip-layout copying case (Micro Law). *Stern, Richard H., M-M Aug 91 3-6, 94*

Legal factors; cf. Copyright protection; Software protection

Local area networks

multibackend database supercomputer interconnected by LAN; architecture and performance. *Hsiao, David K., M-M Dec 91 44-60*

M

Machine vision

2D analog VLSI network simulating function of human visual, peripheral processes. *Li, Hua, + , M-M Oct 91 8-11, 44-47*

Management; cf. Project management

Mass memories

trends in hardware and applications of microcomputers for next ten years. *Myers, Ware, M-M Feb 91 10-13, 68-74*

Measurement

use of English units in European hardware and software systems (Micro World). *Kirrmann, Hubert, M-M Aug 91 36-39*

Measurement-system data handling

Futurebus interface design using off-the-shelf parts for graph reduction in parallel (GRIP) system. *Peyton Jones, Simon L., + , M-M Feb 91 38-41, 84-93*

Measurement-systems data handling; cf. Data buses

Memories; cf. Associative memories; Cache memories; Holographic memories; Mass memories; Microcomputer memories; Optical memories

Memory management; cf. Protocols, memory

Mental models; cf. Cognitive science

Message switching

interprocessor communication performance evaluation of five types of hypercube and grid-topology multicomputers. *Zhang, Xiaodong, M-M Apr 91 12-15, 52-55†*

Microcomputer applications

trends in hardware and applications of microcomputers for next ten years. *Myers, Ware, M-M Feb 91 10-13, 68-74*

Microcomputer interfaces

microcomputer-interfacing laboratory projects. *Fulcher, John A., M-M Feb 91 18-21, 75-78*

Microcomputer memories

trends in hardware and applications of microcomputers for next ten years. *Myers, Ware, M-M Feb 91 10-13, 68-74*

Microcomputer networks; cf. Distributed computing; Multiprocessing

Microcomputer performance

IBM RISC System/6000 architecture and performance; benchmark details. *Oehler, Richard R., + , M-M Jun 91 14-17, 56-62*
redesign of Clipper C400 RISC architecture. *Sachs, Howard G., + , M-M Jun 91 18-21, 74-80*

Microcomputer peripherals

teaching methods for peripheral hardware class and hands-on multitasking lab. *Schultz, Thomas W., M-M Feb 91 30-33, 80-82*

Microcomputer peripherals; cf. Microcomputer interfaces

Microcomputer software

3-D optical architecture. *Louri, Ahmed, M-M Apr 91 24-27, 65-82*

Microcomputers

microcomputer developments during past ten years (Micro World). *Kirrmann, Hubert, M-M Feb 91 42-44*

Microcomputers; cf. Microprocessors**Microprocessor applications; cf. Specific topic****Microprocessors**

- book review; Intel's Official Guide to 386 Computing (Edelhart, M., 1991). *Mateosian, Richard*, *M-M Jun 91* 50-51
- Datawave single-chip multiprocessor for video applications. *Schmidt, Ulrich*, +, *M-M Jun 91* 22-25, 88-94
- DSP-based Petri-net simulation tool. *Di Stefano, Antonella*, +, *M-M Apr 91* 20-23, 56-64
- evolving architectures for microprocessors; overview. *Slater, Michael*, *M-M Feb 91* 96-95
- Gmicro/100 32-b microprocessor based on TRON specifications. *Yoshida, Toyohiko*, +, *M-M Aug 91* 20-23, 62-72
- Hot Chips II: Sweetening the pot (special issue). *M-M Jun 91* 8-29
- i860 microprocessor design and performance. *Atkins, Mark*, *M-M Oct 91* 24-27, 72-78
- IBM RISC System/6000 architecture and performance; benchmark details. *Oehler, Richard R.*, +, *M-M Jun 91* 14-17, 56-62
- IEEE open microprocessor architecture standard P1754 (On the Edge). *Warren, Carl*, *M-M Oct 91* 30-33
- iWarp microprocessor supporting message-passing and systolic communications for multicomputers. *Peterson, Craig*, +, *M-M Jun 91* 26-29, 81-87
- Metaflow architecture using deferred scheduling and register-renaming instruction shelf to manage out-of-order execution. *Popescu, Val*, +, *M-M Jun 91* 10-13, 63-73
- redesign of Clipper C400 RISC architecture. *Sachs, Howard G.*, +, *M-M Jun 91* 18-21, 74-80
- RISC processor for embedded applications within ASIC. *Roberts, Charles E.*, *M-M Oct 91* 20-23, 68-72
- trends in hardware and applications of microcomputers for next ten years. *Myers, Ware*, *M-M Feb 91* 10-13, 68-74

Microprocessors; cf. Microcomputers; Multiprocessing...**Modeling; cf. Petri nets; Simulation****Multiprocessing**

- 3-D optical architecture. *Louri, Ahmed*, *M-M Apr 91* 24-27, 65-82
- DSP-based Petri-net simulation tool. *Di Stefano, Antonella*, +, *M-M Apr 91* 20-23, 56-64
- Futurebus interface design using off-the-shelf parts for graph reduction in parallel (GRIP) system. *Peyton Jones, Simon L.*, +, *M-M Feb 91* 38-41, 84-93
- ITRON-MP adaptive real-time kernel specification for shared-memory multiprocessor systems. *Takada, Hiroaki*, +, *M-M Aug 91* 24-27, 78-85
- KRPP and DSNS superscalar architectures using task and instruction level parallelism. *Fukuda, Akira*, +, *M-M Aug 91* 16-19, 50-61
- RST cache memory design for tightly coupled Clipper-based multiprocessor system. *Prete, Cosimo A.*, *M-M Apr 91* 16-19, 40-52
- Sure System 2000 fault-tolerant computer using hardware and software local redundancies. *Kabemoto, Akira*, +, *M-M Aug 91* 28-31, 73-78

Multiprocessing; cf. Array processing; Distributed computing**Multiprocessing, interconnection**

- interprocessor communication performance evaluation of five types of hypercube and grid-topology multicomputers. *Zhang, Xiaodong*, *M-M Apr 91* 12-15, 52-55†

Multiprocessing, interconnection; cf. Local area networks**Multiprocessors**

- ITRON-MP adaptive real-time kernel specification for shared-memory multiprocessor systems. *Takada, Hiroaki*, +, *M-M Aug 91* 24-27, 78-85

Multitasking

- teaching methods for peripheral hardware class and hands-on multitasking lab. *Schultz, Thomas W.*, *M-M Feb 91* 30-33, 80-82

Muscles, EMG

- myoelectric signal analysis using computer systems and signal processing techniques. *Knaflitz, Marco*, +, *M-M Oct 91* 12-15, 48-58

N**Networks; cf. Multiprocessing, interconnection; Neural networks; Petri nets****Neural networks**

- optical computing overview; relation to neural networks (Software Report). *Kahaner, David K.*, *M-M Feb 91* 53-56

O**Optical computing**

- optical computing overview; relation to neural networks (Software Report). *Kahaner, David K.*, *M-M Feb 91* 53-56
- trends in hardware and applications of microcomputers for next ten years. *Myers, Ware*, *M-M Feb 91* 10-13, 68-74

Optical memories

- 3-D optical architecture. *Louri, Ahmed*, *M-M Apr 91* 24-27, 65-82

P**Packet switching**

- Futurebus interface design using off-the-shelf parts for graph reduction in parallel (GRIP) system. *Peyton Jones, Simon L.*, +, *M-M Feb 91* 38-41, 84-93

Parallel processing

- book review; Superscalar Microprocessor Design (Johnson, M.). *Mateosian, Richard*, *M-M Jun 91* 51, 102
- IBM RISC System/6000 architecture and performance; benchmark details. *Oehler, Richard R.*, +, *M-M Jun 91* 14-17, 56-62
- Metaflow architecture using deferred scheduling and register-renaming instruction shelf to manage out-of-order execution. *Popescu, Val*, +, *M-M Jun 91* 10-13, 63-73
- rate monotonic scheduling algorithm applied to programming real-time systems (On the Edge). *Warren, Carl*, *M-M Jun 91* 34-38, 102

Parallel processing; cf. Multiprocessing; Pipeline processing**Parallel processing, interconnection; cf. Multiprocessing, interconnection****Patents; cf. Software protection****Personal computers; cf. Microcomputers****Petri nets**

- DSP-based Petri-net simulation tool. *Di Stefano, Antonella*, +, *M-M Apr 91* 20-23, 56-64

Pipeline processing

- fine grain architecture for relational database aggregation. *Abdelguerfi, M.*, +, *M-M Dec 91* 35-43
- redesign of Clipper C400 RISC architecture. *Sachs, Howard G.*, +, *M-M Jun 91* 18-21, 74-80

Project management

- project management, site, track, and mobile computers used in construction of England-to-France tunnel (Micro World). *Kirrmann, Hubert*, *M-M Jun 91* 4-6

Protocols

- Futurebus interface design using off-the-shelf parts for graph reduction in parallel (GRIP) system. *Peyton Jones, Simon L.*, +, *M-M Feb 91* 38-41, 84-93

Protocols, memory

- RST cache memory design for tightly coupled Clipper-based multiprocessor system. *Prete, Cosimo A.*, *M-M Apr 91* 16-19, 40-52

Publishing; cf. Copyright protection**R****Real-time systems**

- ITRON-MP adaptive real-time kernel specification for shared-memory multiprocessor systems. *Takada, Hiroaki*, +, *M-M Aug 91* 24-27, 78-85
- rate monotonic scheduling algorithm applied to programming real-time systems (On the Edge). *Warren, Carl*, *M-M Jun 91* 34-38, 102

real-time computing with IEEE Futurebus+. *Sha, Lui, +, M-M Jun 91* 30-33, 95-100

Redundant systems

Sure System 2000 fault-tolerant computer using hardware and software local redundancies. *Kabemoto, Akira, +, M-M Aug 91* 28-31, 73-78

Routing; cf. Multiprocessing, interconnection

S

Scheduling

Metaflow architecture using deferred scheduling and register-renaming instruction shelf to manage out-of-order execution. *Popescu, Val, +, M-M Jun 91* 10-13, 63-73

rate monotonic scheduling algorithm applied to programming real-time systems (On the Edge). *Warren, Carl, M-M Jun 91* 34-38, 102

Search methods; cf. Database systems, searching

Semicustom integrated circuits; cf. Application-specific integrated circuits

Set theory; cf. Fuzzy set theory

Shielding; cf. Wire communication cable shielding

Signal analysis; cf. Biomedical signal analysis

Signal processing; cf. Video signal processing

Simulation

configurable, virtual microprocessor system to simulate and validate process plant designs in classroom. *Russell, David W., +, M-M Feb 91* 26-29

DSP-based Petri-net simulation tool. *Di Stefano, Antonella, +, M-M Apr 91* 20-23, 56-64

Simulation; cf. Circuit simulation

Software; cf. Computer languages; Design automation software;

Microcomputer software; Multitasking

Software, utility programs; cf. Computer interfaces

Software design/development; cf. Data structures

Software education; cf. Computer science education

Software protection

Ashton-Tate Corp. vs. Fox Software, Inc; applicability of patent and copyright laws to software (Micro Law). *Stern, Richard H., M-M Jun 91* 42-46

Lotus Development Corp. vs. Paperback Software International; court's failure to recognize functionality of standardization (Micro Law). *Stern, Richard M., M-M Feb 91* 48-51

Lotus Development Corp. vs. Paperback Software International; effects of courts decision on software industry (Micro Law). *Stern, Richard H., M-M Apr 91* 30-33

Software standards

Lotus Development Corp. vs. Paperback Software International; court's failure to recognize functionality of standardization (Micro Law). *Stern, Richard M., M-M Feb 91* 48-51

Lotus Development Corp. vs. Paperback Software International; effects of courts decision on software industry (Micro Law). *Stern, Richard H., M-M Apr 91* 30-33

Sorting/merging; cf. Database systems, relational

Special issues/sections

database machines. *M-M Dec 91* 6-76

Far East. *M-M Aug 91* 12-31

Hot Chips II: Sweetening the pot. *M-M Jun 91* 8-29

microprocessors in education. *M-M Feb 91* 14-83

Standards

difficulties arising from diversity of methods for obtaining benchmarks (Micro Standards). *Warren, Carl, M-M Apr 91* 34-35

Futurebus interface design using off-the-shelf parts for graph reduction in parallel (GRIP) system. *Peyton Jones, Simon L., +, M-M Feb 91* 38-41, 84-93

Standards; cf. IEEE standards; Software standards

T

Taiwan

growth of computer industry in Taiwan (Software Report). *Kahaner, David K., M-M Jun 91* 39-41

Technology forecasting

trends in hardware and applications of microcomputers for next ten years. *Myers, Ware, M-M Feb 91* 10-13, 68-74

Terminology

book review; Computer Dictionary. *Mateosian, Richard, M-M Aug 91* 43

TV; cf. Video signal processing

Twisted-pair cables

shielded twisted-pair cable characteristics evaluation using Math CAD (On the Edge). *Warren, Carl, M-M Feb 91* 46-47

U

Uncertain systems; cf. Fuzzy set theory

United Kingdom

project management, site, track, and mobile computers used in construction of England-to-France tunnel (Micro World). *Kirrmann, Hubert, M-M Jun 91* 4-6

V

Very-large-scale integration

2D analog VLSI network simulating function of human visual, peripheral processes. *Li, Hua, +, M-M Oct 91* 8-11, 44-47

Gmicro/100 32-b microprocessor based on TRON specifications. *Yoshida, Toyohiko, +, M-M Aug 91* 20-23, 62-72

VLSI accelerators for improving large database system performance. *Lee, Kuo Chu, +, M-M Dec 91* 8-20

Video signal processing

Datawave single-chip multiprocessor for video applications. *Schmidt, Ulrich, +, M-M Jun 91* 22-25, 88-94

Virtual computers

configurable, virtual microprocessor system to simulate and validate process plant designs in classroom. *Russell, David W., +, M-M Feb 91* 26-29

Vision systems (nonbiological); cf. Machine vision

Visual languages

book review; Postscript Language Reference Manual, 2nd edn. (Systems, A.; 1990). *Mateosian, Richard, M-M Apr 91* 28-29

Visualization

configurable, virtual microprocessor system to simulate and validate process plant designs in classroom. *Russell, David W., +, M-M Feb 91* 26-29

VLSI; cf. Very-large-scale integration

W

Wire communication cable shielding

shielded twisted-pair cable characteristics evaluation using Math CAD (On the Edge). *Warren, Carl, M-M Feb 91* 46-47



Richard H. Stern

Law Offices of

Richard H. Stern

1300 19th Street NW,

Suite 400

Washington, DC 20036

Database system copyrights

Databases, or more generally digital information, differ in important ways from the kind of information traditionally protected under copyright law. Protecting databases by copyright law impinges on different interests of plaintiffs and defendants than does protecting books, pictures, and songs—all traditional subjects of copyright law. A sensible legal policy may therefore call for different trade-offs among competing demands of owners, competitors, and users to maximize the benefits to the public—or minimize the public harm—resulting from the operation of the legal system.

Differences from traditional subjects

Commentators¹ have pointed out ways in which databases differ from the traditional subject matter of copyright law.

- Digital information is often more easily and cheaply copied, making appropriation of such information easier and more profitable in the short run than appropriation of traditional works.
- Copyright law regards as significant the distinctions among literary works (words), audiovisual works (pictures, graphics, changing sequences of imagery), and musical works (sound). But these distinctions blur for digital information. It is all bits or bytes stored on tape, disk, or CD-ROM, no matter how you slice it. Why treat use of some bits differently?
- Books can be read by eye. Pictures are also visually perceived. Music or other sound can simply be listened to and apprehended by ear. Most users come equipped with these hardware (or wetware) devices. Databases cannot be used without appropriate special-purpose hardware and software.

- Finally, digital information is more easily modified into another form. You cannot readily turn the *Mona Lisa* into something else. But any time you access a database you can rearrange the downloaded material into almost any desired format. Users can and do readily create new things with old material, making new works that may be eligible for copyright protection. The *Mona Lisa* is in the public domain. So is Beethoven's *Eroica*. With a database and appropriate hardware and software, a user can create a new *Eroic-Mona* pastiche.

Another aspect of database systems, as they are now developing, defies even comparison with the traditional subject matter of copyright. This is the issue of what's the data and what's the program. I do not mean that you cannot inspect bits in a CD-ROM and figure out which bits comprise data and which bits comprise instructions that manipulate the data. You may be able to do this, if someone has provided a debugger or disassembler or reverse compiler; otherwise, probably not. But so what?

Who owns the data?

The problem I mean is that a database user gets results by using a legally protectable computer program to manipulate data that is largely unprotectable in itself. This is because the data is old or simply was not created or owned by the proprietor of the database. Sweat equity in collecting and storing a mass of factual data, such as names and addresses of people who have telephone numbers in a town, is not legally protected under copyright law except for original aspects of selection and arrangement, if any. (The Supreme Court recently so held in a decision that a telephone company held no copyright on its white pages directory, because there was no au-

thorial originality in the content.²⁾

The economic value of the database—and of any product you can create from it—comes from both the data and the software for accessing and organizing it. Yet the resulting product does not visibly contain the software. It just contains the data, in a differently manipulated format. The result is like a compiled program that does not include any “runtime modules.” (Runtime modules are units of code written for a compiler package and incorporated into a compiled code as part of compilation of source code.) Traditional copyright doctrine would suggest there is no computer program in the result and thus nothing legally protected. But that result may not make the best sense from a business standpoint or in terms of encouraging desired forms of enterprise. Opinions will differ.

You might take that example further, in terms of user interfaces. One of the things that makes a database easy to use is a well-designed user interface. In a sense, the interface contributes to the ease of making new products from a database and the quality level they reach, but the new product does not visibly contain the interface itself. Some might consider the resulting legal implications a cause for regret. Others will applaud.

Blurred lines of authorship

Generally speaking, copyright law protects only results that embody authorial originality—some kind of creative spark by a person claiming to have done the creating. This legal principle impinges on databases in two ways. First, the proprietor of the database system, including whatever software is involved, cannot claim to be the creator of what some other person (a user) brings into existence by using the system. Building a soldering iron does not make you the creator of a breadboard whose components someone else solders together.

Second, there may be a question about whether even the person who

uses the database system to produce something is an author of an original work. The person may not contribute enough, either. This may be a situation in which authorship falls into the bit bucket.

The same kind of question has been raised for hypertext linkages for database entries. A hypertext linkage is a facility like a footnote. If this text (for example, the preceding sentence) were on your screen because it belonged to a database that you were using (who knows, *IEEE Micro* on line for hypertext may be just around the corner), you could mouse click on the word “hypertext” (or otherwise enter it). A window would appear, showing a definition of hypertext. Or you could click on the word “footnote” and a window would pop up, explaining what a footnote is. That would occur, however, only if someone went over this text and created a set of hypertext linkages. The linkages would invoke routines that make the program controlling display of this text jump to an entry elsewhere in the database system, to stored text describing hypertext, footnotes, or whatever.

Creating such hypertext linkages is said to be a form of scholarship. Doubt has been expressed, however, that such scholarship results in any copyrightable work of authorship. If that means that no effective reward system will encourage creation of hypertext linkages, perhaps there is a social problem.

On the other hand, creation of some hypertext linkages for my text might offend me. That, too, could cause a social problem. Suppose that someone links this text to a database dictionary entry to exemplify the term “GIGO.” Perhaps, I would not want my work linked in that manner. Should authors have control over hypertext linkages to or from their work? Under traditional copyright law principles, I can prevent my work from being reproduced without permission (subject to fair-use limitations on my rights). Hypertext linkages may dilute that right. Is there a

copyright infringement that I can suppress by law when someone makes a hypertext linkage? Probably not. There is no unauthorized reproduction of my work (once it has lawfully been placed into the database), public performance of it, or other recognized category of copyright infringement.

Those are some of the problems we can anticipate when the courts try to apply copyright law to database systems. It is the same kind of problem you run into when you try to use a set of legal categories developed for one purpose as a mechanism for regulating something else that somebody says ought to be regarded as the same thing and therefore regulated in the same way. What is ever the same as anything else that went before? That has been a problem since the Greeks looked at rivers or tried to address convergence toward a limit. (See box on next page.)

Sometimes this legal process works, because the two things are enough the same for the purpose at hand, or are sufficiently the same considering how much (or little) money you want to spend to resolve the issue. Other times, the process works dismally, because the new subject matter is so vastly different from the old that the legal results do not pass the well-known subjective tests. (These tests have various names, such as the “straight face” test or the “barf” test. The question is whether the judgments that courts reach using the legal theory in question are too ridiculous.)

It would be premature to pass judgment now on whether applying copyright law to the use of database systems will pass or fail these tests. The real-world chickens haven't come home to roost, yet.

References

1. P. Samuelson, “Digital Media and the Changing Face of Intellectual Property Law,” *Rutgers Computer & Technology J.*, Vol. 16, 1990, p. 323.

2. Feist Publications, Inc. v. Rural Telephone Service Co., 111 S. Ct. 1282 (1991).

When are two things the same?

Heracleitus was one of the first to address this issue when he asked if one ever steps twice into the same river. Zeno then came up with a misguided way of addressing a series of the form $t = (s/v)(1 + 0.1 + 0.01 + \dots)$. He believed there was no answer, that is, no knowable value for t , simply because the series was infinite.

Zeno argued that Achilles, running 10 times as fast as a tortoise creeps, could never catch up to it. Each time Achilles closed a gap ds , the tortoise advanced another increment of $0.1 ds$, and thus would forever stay ahead. (If Zeno believed that, he would have been willing to believe a great many other things.)

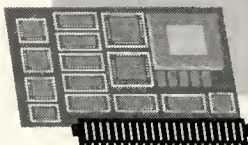
This is much like the logic of copyright law and indeed legal logic in general. Which parameters you designate as independent variables determine the outcome. If you decide to talk about $F(g)$, the plaintiff wins; if you decide to talk about $G(f)$, the defendant wins. It's like integrating by parts, where $duv = u dv + v du$. What you decide to designate u and what you decide to designate v (or dv) determines whether you will solve the problem or just make it worse.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card

Low 177 Medium 178 High 179

On the Edge



Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 6-0669

warren@ssdunix.mdc.com

SBus: an open bus architecture

[In the third part of our series on tools, Rudolf Usselmann discusses the SBus and its progress towards IEEE standardization.]

I invite readers to send information on a tool or method that solves problems, for consideration in future columns.—C.W.]

Rudolf Usselmann
Sparc International

Sun Microsystems originally developed the SBus for the Sparcstation 1 as an inexpensive, high-performance system interconnection bus. "The original goal of the SBus was to provide an expansion bus for free, as our cost target for the Sparcstation 1 was zero dollars," said Jim Ludemann of Antares Microsystems. Ludemann was one of the original codesigners of Sparcstation 1 and SBus.

"This meant we had to design a bus flexible enough for I/O, but fast enough to connect main memory with the cache," Ludemann said. "The SBus was literally the only bus in Sparcstation 1. We knew that in future system implementations the SBus would be used only for I/O, as system design considerations require a custom memory bus for each new system. Furthermore, SBus was designed from the start to be implemented with CMOS gate arrays,

giving us low cost, high performance, and low power usage."

As we can see in many implementations of SBus, the company followed its plans precisely. Even clone manufacturers followed these steps and implemented SBus in their systems. A good example is the Toshiba Laptop, which has one SBus slot for extensions.

Features

SBus offers a wide variety of advantages for system and peripheral developers that other buses cannot offer as a package. Individually, its features are easily achieved and are incorporated in other buses. As a combined package, they offer a very good solution for modular I/O subsystems extensions. (See Table 1 on page 81 for detailed signal definitions.)

Low device count. Various companies offer one-chip solutions for SBus developers. These chips integrate the complete SBus interface on one IC. Peripheral developers have an easy task connecting standard I/O devices (serial I/O, SCSI, and LAN controllers) to single chips and don't have to worry about correct SBus implementation. System developers can purchase interface chips that allow communication from high-speed CPU interconnection buses (for example, MBus) to SBus, thus simplifying the task of system development dramatically.

Low-power CMOS implementation. SBus is 100 percent compatible

with CMOS technology, which eliminates the need for high current drivers and large power supplies, as well as problems with heat dissipation.

Flexible high performance. The original implementation of SBus allows data transfer rates up to 80 Mbytes/s. After the company released SBus to public domain, users formed an SBus working group and started working on an extension.

The SBus public committee developed an extension for SBus to allow 64-bit operations. This meant defining new transfer types and introducing modes without breaking current implementations. It was not a simple task.

The extension of SBus to 64 bits increases performance to more than 160 Mbytes/s. This will help I/O performance keep up with ever-increasing CPU performance. In the past, many I/O devices could not connect to an I/O bus because of their need for high throughput. The extensions will allow

these devices to connect to SBus. However, SBus sizing does not limit one to a certain transfer rate. SBus dynamic bus sizing and a wide variety of cycle types can fit many needs.

Open Boot. Open Boot firmware supports the SBus and assists software drivers. Each SBus board has a small ROM containing identification information and optional boot and diagnostic drivers. The identification information includes such generic items as the device name, register addresses, and interrupt levels and may include device-specific items such as the resolution of a display device.

This ROM information is stored as a computer program written in F Code, a byte-coded version of the Forth programming language. F Code is independent of the CPU's instruction set architecture, so the SBus device's F Code ROM can be used without change on systems with different CPU types.

The Open Boot firmware residing on

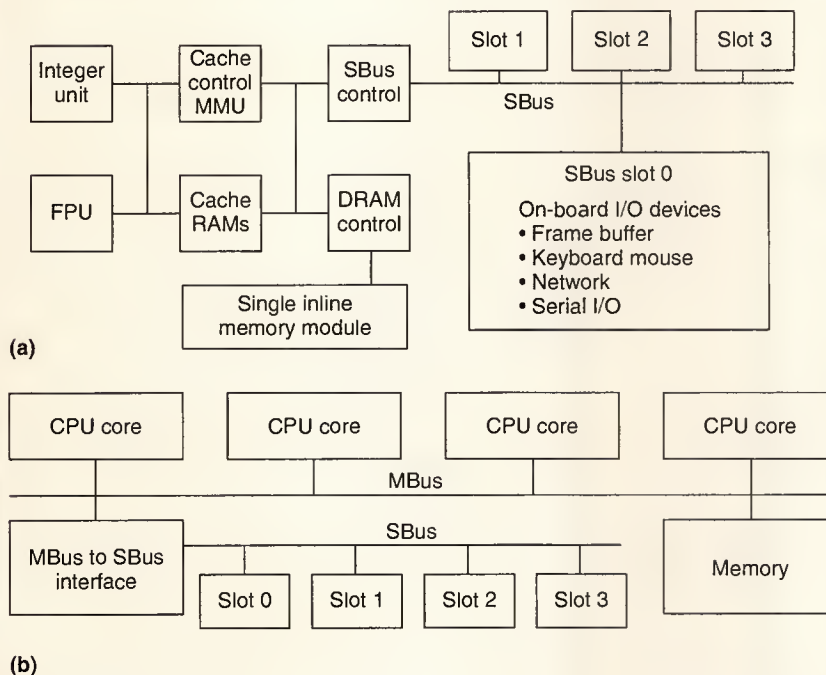


Figure 1. Low-cost (a) and high-performance (b) systems.

Contacts

SBus study group

e-mail: sbus_sg@sparc.com
(Mail goes to 150 members)

Wayne Fischer
Chair, SBus study group
Force Computers
3165 Winchester Blvd.
Campbell, CA 95008
phone: (408) 370-6300
fax: (408) 374-1146
e-mail: uunet!force-clwfischer

SBus study group alias maintenance and requests

e-mail: sbus_sgreq@sparc.com

SBus Specifications

Hamilton Avnet Electronics
(800) 442-6458

Sun's Interoperability Center

Yatin Trivedi
2550 Garcia Ave., PAL1-106
Mountain View, CA 94043
(415) 336-1812
e-mail: trivedi@sun.com

the CPU board contains an interpreter for the F Code language. Interpretation of the F Code ROM creates an entry for the device in the firmware's device tree, a hierarchical data structure describing the system configuration. The operating system software inspects this device tree to determine which devices are present in the system and the individual characteristics of those devices.

If the F Code ROM contains diagnostic and/or boot drivers, those drivers may be used by the firmware for such tasks as testing the device hardware, loading the operating system from that device, or displaying start-up messages using the device.

Open Boot's F Code scheme is not limited to SBus. F Code is being considered for use with Futurebus+ and

Table 1. Signal definitions for SBus standard 32-bit and extended 64-bit modes. All signals are distributed via a high-density 96-pin connector, which also includes GND, +5V, and $\pm 12V$ supplies.

Signal	I/O	Driven by	Description
PA(27:00) [D(59:32)]*	I	Controller	Physical address
Sel_	I	Controller	Slave select (one per slave)
D(31:0) [D(31:0)]*	I/O	Masters/slave	Data
Siz(2:0) [D(62:60)]*	I/O	Masters	Transfer size
Rd [D(63)]*	I/O	Masters	Transfer direction
AS_	I	Controller	Address strobe
Ack(2:0)_	I/O	Slaves/controller	Transfer acknowledgment
LErr_	I/O	Slaves	Late data error
BR_	O	Masters	Bus request (one per master)
BG_	I	Controller	Bus grant (one per master)
Clk	I	Controller	SBus clock
Reset_	I	Controller	Reset
IntRes(7:1)_	O	Slaves	Interrupt request (open drain)
DtaPar	I/O	Master/slaves	Data parity (optional)

In an extended mode, the signals below are time multiplexed

Signal	Description
D(31)	Extended transfer type
D(30:28)	Extended transfer size (2:0)
D(27)	Extended transfer read
D(26:25)	Extended transfer atomic (1:0)
D(24:0)	Extended transfer reserved (24:0)

Supported sizes

Signal	Description	Description
ETSiz(2:0)		Extended (64-bit) mode
Siz(2:0)	Standard (32-bit) mode	
000	Word (4-byte) transfer	Reserved
001	Byte transfer	Reserved
010	Half-word (2-byte) transfer	Reserved
011	Extended Transfer	8 bytes
100	Four-word burst (16 bytes)	16 bytes
101	Eight-word burst (32 bytes)	32 bytes
110	Sixteen-word burst (64 bytes)	64 bytes
111	Two-word burst (8 bytes)	128 bytes

Acknowledgment encoding

Ack(2:0)_	Function
111	Idle/Wait
110	Error acknowledgment
101	Byte (data) acknowledgment
100	Rerun acknowledgment
011	Word (data) acknowledgment
010	Double-word (data) acknowledgment
001	Half-word (data) acknowledgment
000	Reserved

Atomic cycle types

ETAtomic(1:0)	Description
00	Normal bus cycle (non-atomic bus cycle)
01	First bus cycle of an atomic transaction
10	Intermediate bus cycle of an atomic transaction
11	Last cycle of an atomic transaction

* Signals in square brackets are for 64-bit mode. They are shared with the regular signals when that mode is enabled.

_ Signals with an underscore mark indicate low-active signals.

VME-D, and its design applies to other buses as well. Open Boot firmware is the subject of an IEEE standardization effort under project number P1275.

Current status

Sun Microsystems has placed SBus in the public domain and helped to establish a Public Specification Com-

mittee to continue developing the SBus specification. This committee is working on extensions to SBus and is addressing the undefined and unclear

aspects of its specification.

Ongoing project

Because of the wide acceptance of the SBus standard, Wayne Fischer of Force Computers asked the IEEE Computer Society and the Bus Architecture Standards Committee to open a Project Authorization Request (PAR) to establish SBus as an IEEE standard. In June 1991 IEEE's Bus Architecture Standards Committee agreed to sponsor a study group. At the June meeting of the SBus Public Spec Committee, Fischer suggested this group merge into the SBus study group. The committee unanimously approved. At publication, this group has met three times, most recently on December 3-4. The next meeting is scheduled for January 22-23, 1992, in Salt Lake City.

Interoperability Center

The Interoperability Center offers SBus card developers help in designing hardware and software and assists with porting SBus subsystems to the company's different platforms. It is equipped with most hardware and software tools a designer might need, including logic analyzers, oscilloscopes, and most of the popular schematic editors available on the market. The

center has a well-experienced staff to assist designers with problems.

Compatibility

Today vendors produce more than 400 different SBus boards. Thus, compatibility is becoming an increasingly important issue to vendors and end users. Sparc International introduced, in summer 1991, a program for compatibility testing of SBus boards on Sparc-compliant platforms. This service verifies the compatibility of SBus boards and accommodating software across a wide variety of platforms.

In a joint effort with VME Labs, Sparc International plans to expand this program to include full system-independent hardware and software verification by mid 1992. VME Labs plans to perform hardware verification; Sparc International plans to test device drivers and other accommodating software. Currently the lack of a device driver standard makes testing impossible. However, Sparc International and its members are developing a uniform standard for the next Unix release SVR4.

Summary

Considering the typical lifetime of an I/O bus, SBus is in its infancy and has a lot of room to grow. If it becomes a

standard, its growth will be guaranteed in a controlled manner, not in chaos, like other buses. Furthermore, the standardization of SBus will ensure its continued support and expandability.

Acknowledgments

I thank Jim J. Ludemann of Antares Microsystems, Mitch Bradley of Sun Microsystems, and Wayne Fischer of Force Computers and the SBus study group for their help with this article.

Rudolf Usselmann, a senior architect at Sparc International in Menlo Park, Calif., develops extensions to the Sparc architecture and related products such as the SBus and MBus and performs compliance testing on Sparc processors and peripherals.

Usselmann holds a Diploma in computer science and electrical engineering from the University of Hamburg.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card

Low 192 Medium 193 High 194

CALL FOR PAPERS!

SPECIAL ISSUE, IEEE DESIGN & TEST OF COMPUTERS ON DESIGN AND TEST OF MEGABIT MEMORIES

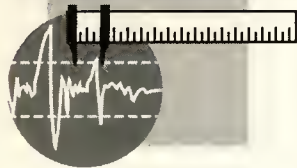
IEEE D&T will devote its March 1993 special issue to a wide range of topics on the design and test of high-density and high-speed memories. The memory design process that in the past had been the domain of circuit and process development experts now requires expertise in various other fields to implement testability features and device architectures for high-speed access using pipelining, interleaving, and so on.

DEADLINES

Submit six copies of a 20-page, double-spaced, typewritten paper **before April 1, 1992**. You will be notified by September 1, 1992, if your paper is accepted. Final version of your paper must arrive no later than November 1, 1992.

Send submissions to Manuel d'Abreu, editor-in-chief, GE Corporate R&D, 1 River Rd., Bldg. KW-C308, Schenectady, NY 12301; (518) 387-7097; dabreu@crd.ge.com. Direct any questions to either the editor-in-chief or guest editors Pinaki Mazumder or John P. Hayes, Dept. of EE&CS, University of Michigan, Ann Arbor, MI 48109-2122; (313) 763-2107 or (313) 763-0386; mazum@indus.eecs.umich.edu or jhayes@eecs.umich.edu.

Micro Standards



Computing interconnections

The emphasis on networking puts a new burden on system designers and software engineers: the interchange of data in a seamless fashion. Many methods have been considered and have proven workable at least on a one-to-one basis. For example, early on *Byte* magazine attempted to provide a common format for information interchange using the Kansas City Standard for audiotape recording of computer data and programs. The system worked in the context of the machines and users at that time.

Publishers of the now-defunct *Interface Age* magazine, of which incidentally I served as editor-in-chief, saw in 1977 the need for ensuring that common data was used in the many different processor types and operating systems. The solution was the IPIS (International Platform Interchange System). This approach consisted of a software process that ensured that data either conformed to a specific format or could be translated to that format with some manipulation on a host or target system. The problems with IPIS were that no ground swell requirement for interchange existed and operating systems and languages were still being defined. Essentially we had a solution that was out of context with the need at the time. We had come up with an interesting addition to computer science, but it didn't catch on.

Robust systems

Today, manufacturers deliver platforms with robust and well-defined operating systems and environments. It isn't unusual to find a large network consisting of Macintoshes, IBM PCs, Sun and Hewlett-Packard workstations, and large mainframes. In some cases the ability to interchange data is common due to the operating

environment. Unix machines, for example, provide not only a common file transport mechanism (File Transport Protocol—FTP or Network File System—NFS) but also file and data formats that are similar if not exactly the same.

Not all systems run Unix, and not all Unix boxes are implemented in the same way. Consequently, not all data can be guaranteed to translate to another platform the same way. A Unix-to-VAX/VMS interchange not only requires translating syntax and methods of naming files but the basic file structures as well.

To this end, work has been going on to develop a common interconnection tool that ensures the easy and obvious way of moving and sharing data.

The interconnection tool

The IEEE effort is P1175/D11 (draft 11), A Standard Reference Model for Computing System Tool Interconnections, prepared by the IEEE Computer Society's Task Force on Professional Computing Tools. The balloting committee has approved P1175, which awaits final approval by the IEEE Standards Board this month. For copies of P1175, telephone the IEEE Standards office in Piscataway, New Jersey, at (800) 678-IEEE; the fax number there is (908) 981-9667. For answers to questions about the draft standard, contact P1175 Chair Robert M. Poston, Programming Environments, Inc., 4043 State Highway 33, Tinton Falls, NJ 07753; telephone (908) 918-0110 or fax (908) 918-0113.

The P1175 document describes a robust solution to resolving interconnection problems called the Semantic Transfer Language (STL). The standard describes STL as parsable, easy to read and understand by programmers, easy to write, and easy to convert to a compact transfer form. The

Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 6-0669

warren@ssdunx.mdc.com

notion of the STL describes items in the form of English sentences that contain one subject, a verb or verb phase, and only one relation or attribute to the subject. To maintain commonality with existing systems and languages, STL uses a modified version of the Backus-Naur Form (BNF) for syntax description. For example, ::= means "is defined as," and white spaces or tabs mean "is followed by." Similarly, STL relates items by concepts and actions or functions of the concepts, such as those listed in the box.

Interestingly, the overall concept of P1175 and STL is based on communications concepts overlaid onto system and software methods. Within the software constraints the issues of concern are actions, transformation of state, control and data; events, control, timing, and synchronization of actions; information and data, the primary object of software actions; logic, logical restrictions (conditions) on actions; and finally states, the context for and evolution of actions.

Standard not limited

The architects of P1175 haven't limited the standard by looking only at a narrow range of hardware and software concepts. Rather, they began at the uppermost part—the user hierarchy, the organization. They thought that if information can't be easily and readily shared by an organization, it isn't useful. Moreover, the organization establishes the operation context and defines the platform(s) that will be used. For example, McDonnell Douglas primarily uses Macintosh IIcx systems and VAX mainframes, while supporting subcontractors using IBM PCs and a variety of other platforms.

The next issue addressed by the working group was tool-to-platform interconnections. This so-called architectural context is guided by the organization context. When the organization is closed, the process is simple, and platform-to-platform interconnections work easily. However when the

mix is great either within or external to an organization, the problem becomes greater, and tools must be available for interconnecting platforms. The most logical way to solve the problem is to use networks that range from local area networks (LANs) to wide area networks (WANs).

Once the organization and architectural issues and interconnections were worked out, the committee tackled the problem of interconnections among tools, the transfer context. How does data element A on platform A probably get to platform Z over the network and remain a useful entity? Enter STL. The Semantic Transfer Language provides the mechanism to assist in the translation—not a trivial concept.

Not a lonely effort

The IEEE effort isn't one that was pursued without support. Indeed, many members of the task team represent organizations and other standards groups that are interested in information interchange and interconnections. One notable effort that has been tak-

ing place resulted from NASA's Space Station Freedom. The space station will be using a processing system to collect and download sensor data to earth stations. This system brought about the development of a Data Naming Standard and a translation tool for ensuring that all the associated databases function in a common way. Thus the schema (the structure of the database) of one database can effectively communicate with another database that has a different purpose.

With this in mind and using communications concepts, the company developed the Interdiscipline Systems Definition Encyclopedia (ISDE). This tool allows us to define database schemas in a common way and provides a translation mechanism so that any database can be made to appear to conform to a common format. ISDE is implemented as a database for the Macintosh using 4th Dimension Database Manager software from Acius Corp. The software checks schemas against the McDonnell Douglas Data and Object Standards (DAOS) naming standard for conformity.

Although McDonnell Douglas designed ISDE to solve a specific problem associated with the space station, it arranged ISDE to follow the same rules as P1175. However, the approach is one of a relational database rather than a semantic transfer language. Both methods have merit and should be combined. Therefore, I'm recommending to the P1175 chair that the ISDE model be considered as an addendum to the basic P1175 work. Combining the STL with a relational database tool will result in a common tool definition that will ensure accurate information interchange regardless of platform or implementation. However, although I will seek a Project Authorization Request for this work, P1175 first faces a two-year trial-use period (beginning shortly) before an addendum can be considered.

For additional information about the Data and Object Naming Standard

STL concepts

- Action
- Collection
- Condition
- ConnectionPath
- Constant
- DataItem
- DataKey
- DataPart
- DataRole
- DataStore
- DataType
- DataView
- EventItem
- EventType
- GraphicSymbol
- Object
- S_Packet
- State
- StateTransition

(DAOS), contact C.R. Easton, McDonnell Douglas Space Systems Co., Space Station Div., 5301 Bolsa Ave., M/S A95-J845-17/6; Huntington Beach, CA 92647; telephone (714) 896-4551; fax (714) 896-5034.

For more information about ISDE, contact Lee Neitzel, CTA Corp., 18333 Egret Bay Blvd., Suite 310, Houston, TX 77058; telephone (713) 333-2436; fax (713) 333-2493.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 180 Medium 181 High 182

1951-1991
 **IEEE Computer Society
Press Books**

MULTIPLE-VALUED LOGIC IN VLSI DESIGN edited by Jon T. Butler

This book provides a historical perspective on MVL, focuses on various technologies for implementing multiple-valued VLSI circuits, delves into applications of MVL in diverse technologies, and discusses device physics, logic characteristics, and small and medium-scale circuit experiences. The text contains 12 papers divided into four categories—Introduction, Multiple-Valued Logic Technology, Special Implementations, and Multiple-Valued Logic Tools and Techniques.

128 PAGES. JULY 1991. SOFTBOUND.
ISBN 0-8186-2127-3.
CATALOG NO. 2127 \$35.00
MEMBERS \$28.00

**Call toll-free
to order
1-800-CS-BOOKS
(in CA (714) 821-8380)**



Joe Hootman

University of North Dakota

Send announcements of new microcomputer and microprocessor products to
Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264

Workstations

Entry- and mid-level workstations

Users who have outgrown their personal computers but aren't ready to move up to high-end workstations may want to consider Sun's IPX and EPX Sparcstations.

The 16-Mbyte IPX (expandable to 64 Mbytes) is a mid-level workstation designed for a variety of applications, including complex, computer-aided software development, financial analysis, electronic publishing, and network file/database access. It achieves 28.5 MIPS and 4.2 Mflops with a rating of 24.2 Specmarks. GX accelerated graphics speed window response time and allow users to manipulate multiple windows and complex objects in near-real time. IPX works as a single unit or configures as a file server for small work groups.

The ELC yields 21 MIPS and 3 Mflops with 20.1 Specmarks. It offers multiple windows, a high-resolution display, fast response times, built-in networking, and 8 Mbytes of standard memory (expandable to 16 Mbytes). It also configures as a file server. ELC integrates all of the workstation components into the back of a monochrome display. *Sun Microsystems*; \$11,995 (IPX), \$4,995 (ELC).

Reader Service No. 10

25-MHz SBus Sparc

Compstation 25 is an SBus-compatible, Sparc-compliant definition 1.1 workstation that runs on Sparc/OS 1.1.1. It yields 15.8 MIPS and 1.75 Mflops with 10.25 Specmarks. The system features a high-resolution, 19-in. color monitor and 8 Mbytes of RAM (expandable to 64 Mbytes). It supports Sun View, Open Windows, and Motif software. An SBus add-on card allows users simultaneous access to Unix and DOS environments. *Tatung Science and Technology*; \$7,995.

Reader Service No. 11

16-Mbyte Sparc CPU

The power and software capability of a Sun workstation running SunOS is available to the embedded systems market in a 16-Mbyte Sparc CPU. The Sparc CPU-1E/16 is a VME single-board computer that offers enough on-board memory to run standard SunOS in a single-slot environment without a memory expansion board. It achieves 12.5 MIPS and 1.4 Mflops with 8.4 Specmarks. *Force Computers*; \$7,995.

Reader Service No. 12

Computer boards and cards

CMOS 68000 single-board computer

Designed for use in remote or battery-powered portable applications, the

MPL-4079 board consumes 120mA with the CPU running at 8 MHz. Six 32-pin JEDEC sockets can be equipped with up to 512 Kbytes of PROM and 256 Kbytes of battery-backed RAM on a 25-sq in. CMOS RAM.

The device also has two RS-232 serial ports, a battery-backed clock calendar, two 16-bit timers, and a four-channel, 8-bit, A/D converter. The board's G-64 bus interface expands I/O capabilities. It operates from -40° to +85° C. Software for the MPL-4079 can be developed on DOS and downloaded to the board's memory using Crosscode C and PC bridge cross development software packages. *Gespac; \$595 (100s).*

Reader Service No. 13

PCs can run Macintosh software

DOS users can install the Andor One add-in board with associated software to run Apple Macintosh software. The chip's circuitry allows a DOS machine's 3.5-in. drive to directly read and write Macintosh disks. The full-length PC card installs in any PC, from the earliest PC-XTs through 486-based machines. Andor One has terminate-and-stay-resident software that takes 60 Kbytes of RAM. The package also includes Word-for-word/Mac, a DOS-Mac translator.

According to the company, Andor One runs twice as fast as a Macintosh Classic. It is compatible with DOS-standard peripherals such as mouse, keyboard, hard disk, 3.5-in. disk drives, and monitors. An Apple Talk-compatible RS-422 connector allows it to network with Apple Laserwriters, Apple Local Talk, Phonenet, and other networking devices. The board requires Mac-plus ROM and Macintosh Systems and Finder software. *Hydra Systems; \$995.*

Reader Service No. 14

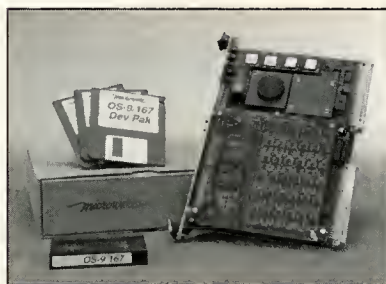
Enhanced OS-9 system

OS-9/MVME167 Development Pak is an optimized version of the OS-9 real-

time operating system. Its manufacturer says it takes full advantage of the power of Motorola's 32-bit M68040 processor while providing support for the advanced on-board serial, SCSI, and Ethernet hardware.

The package includes a number of OS-9 device drivers for I/O peripherals included on the MVME167 board family, including SCSI drivers for the NCR 53C710 controller. Other drivers support the on-board, real-time clock and the CD-2401 serial I/O controller. An Internet Support Package supports Ethernet. Also included are a set of resident development tools for application development, including K&R C compiler, macro assembler and linker, user state debugger, μMACS full-screen editor, and shell command interpreter. The system also comes in a modified Run-Time Pak that excludes device drivers and development tools. *Microware Systems; \$3,000 (Development Pak), \$1,500 (Run-Time Pak).*

Reader Service No. 15



Microware's OS-9 Development Pak

Classic runs twice as fast

The Classic Performer boosts the speed and performance of the Macintosh Classic by as much as 96 percent, its manufacturer says. The accelerator board is a 68000 processor that runs at 16 MHz, twice the clock speed of the Classic's built-in processor. It has a 64-Kbyte SRAM cache circuit and 25-ns PAL chips. The manufacturer says it speeds math calculations by up to 75 percent and accelerates the SCSI port by 15 percent. An optional math

coprocessor boosts math-intensive programs by as much as 5,000 percent. It supports Macintosh systems above 6.0.7. *Harris Laboratories; \$299.95 (Classic Performer) \$149.95 (68881 math coprocessor).*

Reader Service No. 16

Communications software

Data compression for micro-to-mainframe conversion

Version 2 of Compress is an advanced bit compression/decompression utility that processes text and binary files and offers ASCII/EBCDIC conversion tables. The manufacturer says it reduces size and transfer time of text files by 50 to 80 percent.

Compress is compatible with most micro-to-mainframe products with file transfer capabilities. Version 2 also supports OS/2 and Macintosh workstations. It is available in VM/CMS, DOS/VSE CICS, and MVS (TSO and CICS) versions. *Telepartner International; from \$6,000. Free upgrades for customers on maintenance.*

Reader Service No. 17

Network management

The two products in the Network Control Series offer managers the ability to control their enterprisewide networks at central and local levels. Cornerstone Agent allows subnet managers to monitor traffic on the network, filter data, generate statistics, and decode protocols. It identifies potential LAN problems or interfaces with Foundation Manager to allow a central administrator to troubleshoot any LAN on the network. Foundation Manager has in-depth analysis and management capabilities including statistical analysis, network characterization, simulation, auto-baselining, protocol analysis, and network visual mapping.

Both products are based on a graphically programmable user interface and built-in start-up programs. The series supports MIB I, MIB II, and Remote

Network Monitoring MIB. It also supports Token Ring source routing and IP routing. *Pro Tools*; \$8,995 (*Foundation Manager*), \$1,295 (*Cornerstone Agent*). Upgrades for \$500.

Reader Service No. 18

VAX-to-IBM connection

VAX users can connect directly to IBM's LU6.2/Token Ring with EZ Bridge. This software allows VAX users on PCs or VT terminals to log on to a mainframe as a 3270 user and access mainframe applications such as CICS and TSO. VAX users can also print DOS-format files on their local printers and have peer-to-peer communications with LU6.2 systems.

The EZ Bridge family includes three products. The SNA/3270 connects VAX to mainframe by emulating an IBM 3174 or 3274 cluster controller, 3278/9 terminals, and 3287 printers. The SNA/LU6.2 implements IBM's LU6.2 protocol that provides peer-to-peer communications capabilities on an SNA/Token Ring network. EZ Bridge OLTP allows users to access and update data in different locations. *Systems Strategies*; \$3,000 to \$15,000, based on VAX size.

Reader Service No. 19

Communications hardware and chips

Multimedia communications chip

Fax, data, voice, and caller ID capabilities are combined on an LSI chip, the Fax Vodem (YTM403). The chip enables users to send and receive graphics, data, and voice messages on a single line and incorporate caller ID if desired. It supplies voice quality with resolution up to 12 bits at 9,600 bps and dynamic range up to 72 decibels.

A built-in voice record-and-playback capability provides a 12-to-4 bit, three-to-one compression and decompression ratio. An internal analog-to-digital/digital-to-analog converter supports voice-mail capability. A caller ID func-

tion allows receivers to identify the sender before answering the phone or receiving a communication. The chip comes in 64-pin SDIP or 64-pin QFP versions. *Yamaha Systems Technology*; \$40 (1,000s).

Reader Service No. 20

10 Base T through AUI port on Ethernet

The Model 177 transceiver attaches to Ethernet equipment to upgrade LAN coaxial Ethernet adapters and wiring networks to the 10 Base T wiring standard. It uses unshielded, twisted-pair wiring and connects to Ethernet through an AUI port or via an Ethernet transceiver cable.

The unit has four status LEDs indicating link, collision, receive, and transmit. Power derives from the AUI port, making an external source unnecessary. The 10 Base T port uses an RJ45 connector. *Telebyte Technology*; \$119, discounts for quantities.

Reader Service No. 21



Telebyte's Model 177 transceiver

Chips and components

3V CMOS EPROMs

The 27LV256 and the 27LV512 are 3V CMOS EPROMs designed for battery-powered applications in which 5V devices are no longer preferred. The

manufacturer says the 3V EPROMs provide 200-ns access times.

The 27LV256 and 27LV512 are available in speeds of 200, 250, and 300 ns. Both devices come in plastic dip, PLCC, and SOIC packages. *Microchip Technology*; 27LV256 from \$4.73, 27LV512 from \$8.03 (1,000s).

Reader Service No. 22

Hardware-based fuzzy-logic controller

The NLX230 is a single-chip fuzzy-logic inference engine with on-chip rule memory. The manufacturer says it processes 30 to 40 times faster than comparable software-based or hardware/software hybrid fuzzy-logic control solutions. The manufacturer recommends it as a replacement for conventional PID controllers, sequencers, state machines, and intelligent timers.

The controller is housed in a 40-pin package. It supports eight inputs, eight outputs, and a resident base of 64 fuzzy-logic rules. The manufacturer also offers an applications development system that includes the chip and associated circuitry on a PC-compatible board, controlling software, and documentation. *Neura Logic*; \$4 (production quantities), \$395 (development system).

Reader Service No. 23

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 189 Medium 190 High 191

Advertiser/Product Index

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Northwest Region: D. Rodney Brooks; Tel: (415) 905-0260; Fax: (415) 896-1512.

Eastern Region: Georgette Boone; Tel: (415) 905-0260; Fax: (415) 896-1512.

Southwest Region: D. Rodney Brooks; Tel: (415) 905-0260; Fax: (415) 896-1512.

Recruitment and Classified Advertising: D. Rodney Brooks; Tel: (415) 905-0260; Fax (415) 896-1512.

Director of Sales: Randall L. Stickrod, 544 Second St., Suite 200, San Francisco, CA 94107; Tel: (415) 905-0260; Fax: (415) 896-1512.

For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

IEEE MICRO, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

RS # Page

Force Computers	12	86
Gespac	13	87
Harris Laboratories	16	87
Hydra Systems	14	87
IEEE CS Membership	—	21
Microchip Technology	22	88
Microware Systems	15	87
Neura Logix	23	88
Pro Tools	18	88
Sun Microsystems	10	86
Systems Strategies	19	88
Tatung Science and Technology	11	86
Telebyte Technology	21	88
Telepartner International	17	87
Yamaha Systems Technology	20	88



Have you heard about our...

Technical Committee on Microprocessors and Microcomputers

*For information on this, or any of our more than
30 technical committees, contact:*

IEEE COMPUTER SOCIETY
Membership/Circulation Dept.
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1264
(714) 821-4010

THE FOLLOWING INFORMATION IS AVAILABLE:

Contact the Publications Office; to facilitate handling, please request by number.

- Membership application, student #203, others #202
- Periodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Compmail electronic mail brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures #193
- Student scholarship information #192
- Volunteer leaders/staff directory #196
- IEEE senior member grade application #204

(requires ten years practice and significant performance in five of those ten)

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

PURPOSE

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

MEMBERSHIP

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

1951-1991

40 YEARS OF SERVICE



IEEE COMPUTER SOCIETY

A member society of the
Institute of Electrical and Electronics Engineers, Inc.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and product reviews.

Periodicals. The society publishes six magazines and five research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

OMBUDSMAN

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

EXECUTIVE COMMITTEE

President: Duncan H. Lawrie*
University of Illinois
Dept. of Computer Science
1304 W. Springfield
Urbana, IL 61801
(217) 333-3373

President-Elect: Bruce D. Shriver*
Past President: Helen M. Wood*

VP, Standards: Paul L. Borrill (1st VP)*
VP, Press Activities: Barry W. Johnson (2nd VP)*
VP, Conferences and Tutorials: Laurel V. Kaleda†
VP, Education: Raymond E. Miller†
VP, Membership Activities: Ronald O. Williams†
VP, Publications: Ronald G. Hoelzeman†
VP, Technical Activities: Mario R. Barbacci*

Secretary: James H. Aylor*
Treasurer: Joseph Boykin†
Division V Director: Edward A. Parrish, Jr.†
Division VIII Director: Helen M. Wood*
Executive Director: T. Michael Elliott†

*voting member of the Board of Governors
†nonvoting member of the Board of Governors

BOARD OF GOVERNORS

Term Expiring 1991:

P. Bruce Berra, Michael Evangelist,
Ted Lewis, Raymond E. Miller, Earl E. Swartzlander, Jr.,
Joseph E. Urban, Thomas W. Williams

Term Expiring 1992:

James H. Aylor, Alicia I. Ellis, Tadao Ichikawa,
C.V. Ramamoorthy, Sallie V. Sheppard,
Harold Stone, Akihiko Yamada

Term Expiring 1993:

Fiorenza Albert-Howard, Jon T. Butler, Michael C. Mulder,
Yale N. Patt, Anneliese von Mayrhauser,
Benjamin W. Wah, Ronald Waxman

Next Board Meeting

February 28, 1992, 8:30 a.m.
Cathedral Hill Hotel, San Francisco, CA

SENIOR STAFF

Executive Director: T. Michael Elliott
Publisher: H. True Seaborn
Director, Conferences and Tutorials: Anne Marie Kelly
Director, Finance and Information Services: Tod S. Heisler
Director, Board and Administrative Services: Violet S. Doan
Assistant to the Executive Director: Sandra K. Pfau

COMPUTER SOCIETY OFFICES

Headquarters Office

1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone: (202) 371-0101
Fax: (202) 728-9614

Publications Office

10662 Los Vaqueros Cir.
PO Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information:
(714) 821-8380
Publication Orders: (800) 272-6657
Fax: (714) 821-4010

European Office

13, Ave. de L'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05

Asian Office

Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 3408-3118
Fax: 81 (3) 3408-3553



IEEE OFFICERS

President: Eric E. Sumner
President Elect: Merrill W. Buckley, Jr.
Past President: Carleton A. Bayless
Secretary: Hugh Rudnick
Treasurer: Theodore W. Hissey, Jr.

VP, Educational Activities: Richard S. Nichols
VP, Professional Activities: Michael J. Whitelaw
VP, Publication Activities: J. Thomas Cain
VP, Regional Activities: Robert T. H. Alden
VP, Technical Activities: Fernando Aldana

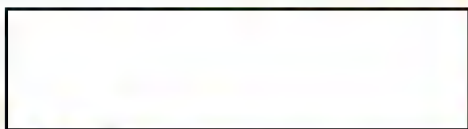
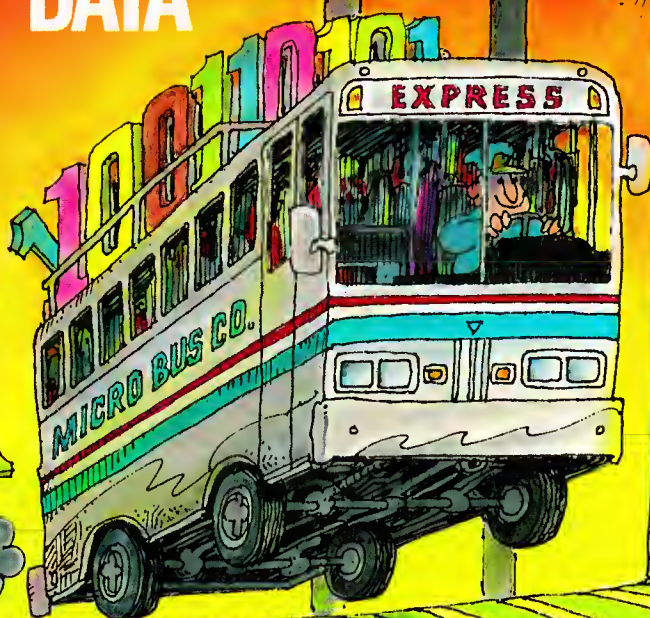
IEEE

MICRO

FEBRUARY 1992

Chips, Systems, Software, and Applications

BUSING DATA



- The Scalable Coherent Interface
- Testing bus products
- Unix and the Am29000
- A neural network classifier
- Hypercube database engine



IEEE COMPUTER SOCIETY



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.



IEEE MICRO

Published by the IEEE Computer Society

February 1992

F E A T U R E S

10 The Scalable Coherent Interface and Related Standards Projects

David B. Gustavson

Avoiding inherent bus problems while providing buslike services that include cache coherence protocols

23 Unix and the Am29000 Microprocessor

Daniel Mann

Assessing the performance of AMD's RISC processor in a Unix system

32 Hardware Requirements for Neural Network Pattern Classifiers: A Case Study and Implementation

Bernhard E. Boser, Eduard Sackinger, Jane Bromley, Yann leCun, and Laurence D. Jackel

Demonstrating special-purpose neural network processor power and flexibility in a hand-written OCR application

42 Experimentation with Hypercube Database Engines

Ophir Frieder, Vijaykumar A. Topkar, Ramesh K. Karne, and Arun K. Sood

Measuring the effects of unique data on parallel-processing performance

57 Conformance Testing of VMEbus and Multibus II Products

Marcus Adams, Yi Qian, Jacek

Tomaszunas, Josef Burtscheidt, Edgar Kaiser, and Csaba Juhasz

Walking through the EC's new automated Bus Interface Conformance Test

*Cover design: Jay Simpson,
Design and Direction*

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; IEEE Computer Society Headquarters, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription: \$23 in addition to IEEE Computer Society or any other IEEE society member dues; \$39 for members of other technical organizations. This journal is also available in microfiche form.

Postmaster: Send address changes and undelivered copies to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Second-class postage is paid at New York, NY, and at additional mailing offices.

Copyright and reprint permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US Copyright Law for private use of patrons those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress St., Salem, MA 01970. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to Permissions Editor, *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Copyright © 1992 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved.

DEPARTMENTS

- 2 **From the Editor-in-Chief**
- 4 **Micro Law**
Disqualifying engineers?
- 7 **Software Report**
Micromachines; forecast for 2010; announcements
- 41 **Editorial Calendar**
- 65 **Micro Review**
Mac System 7 upgrade
- 69 **Micro Standards**
An acronym glossary
- 73 **On the Edge**
Firmware standards
- 75 **Micro News**
Chip density limits; 5- μ m disk laser; natural nanocircuits
- 78 **New Products**
Parallel-processing DSP; R4000 microprocessor; Windows software
- 84 **Product Summary**

*Reader Interest/Service/
Subscription cards, p. 80A;
Advertiser/Product Index, p. 88;
CS information page, cover 3*

IEEE Computer Society
PO Box 3014, Los Alamitos, CA 90720-1264
(714) 821-8380

Editorial: Unless otherwise stated, bylined articles and descriptions of products and services reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society. Send editorial correspondence to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. All submissions are subject to editing for style, clarity, and space considerations.

EDITOR-IN-CHIEF

Dante Del Corso
*Politecnico di Torino**

ASSOCIATE EDITOR-IN-CHIEF

Ashis Khan
*Mips Computer Systems, Inc.***

EDITORIAL BOARD

John Crawford
Intel Corporation

K.E. Grosspietsch
GMD

Joe Hootman
University of North Dakota

David K. Kahaner
National Institute of Standards and Technology

Hubert D. Kirmann
Asea Brown Boveri Research Center

Priscilla Lu
AT&T

Richard Mateosian

Nadine E. Miner
Sandia National Laboratories

Ken Sakamura
University of Tokyo

John L. Schmalzel
University of Texas at San Antonio

Michael Slater
Microprocessor Report

John W. Steadman
University of Wyoming

Richard H. Stern

Philip Treleaven
University College London

Carl Warren
McDonnell Douglas Space Systems Co.

Maurice Yunik
University of Manitoba

STAFF

Marie English
Managing Editor

David Sims
Assistant Editor

H.T. Seaborn
Publisher

Marilyn Potes
Editorial Director

Douglas Combs
Assistant Publisher

Pat Paulsen
Assistant to the Publisher

Jay Simpson
Art Director

Joseph Daigle
Production

Christina Champion
Membership/Circulation Manager

Heidi Rex,
Marian Tilyan
Advertising Coordinators

MAGAZINE OPERATIONS COMMITTEE

John Staudhammer (chair)

Jon T. Butler

B. Chandrasekaran

Carl Chang

Manuel d'Abreu

Dante Del Corso

James J. Farrell, III

John A.N. Lee

Peter R. Wilson

PUBLICATIONS BOARD

Harold Stone (chair)

Ronald G. Hoelzeman

Ming T. (Mike) Liu

Michael C. Mulder

Theo Pavlidis

John Staudhammer

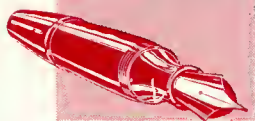
Sallie V. Sheppard

Kishor Trivedi

* Submit six copies of all articles and special-issue proposals to Dante Del Corso,
Dipartimento di Elettronica, Politecnico di Torino,
C.so Duca degli Abruzzi, 24, 10129 Torino, Italy; phone +39 11 564 4044;
Comppmail: d.delcorso; Internet: delcorso@polito.it; Bitnet: delcorso@itopoli
or

** Ashis Khan, Mips Computer Systems, Inc., 950 DeGuigne Drive, Sunnyvale, CA 94086;
Internet: ashis@mips.com.

From the Editor-in-Chief



One year later



NOW THAT I'VE BEEN *IEEE Micro's* EIC for one year, it is time to check the balance sheets; in my introduction of February 1991 I expressed some of my "good intentions." The last two words in the first paragraph in that introduction (I'm sure you keep old issues, so you will know what they were) were surely true.

In this year some things have been accomplished, while others still must be done. I am sure that *IEEE Micro* has been a good channel to bring a lot of useful information to you readers. I myself learned a lot of things. First of all I learned how important the work of the managing editor and staff is to the magazine. I understand now that a magazine could survive without an EIC but not without the staff.

I also learned that Europeans can directly communicate with people in the US only in a narrow time window (and that everybody in the US makes use of an answering machine!). Some big organization should start a project to unify the time on a world basis. (It would be quite easy: You modify the sun into a ring-shaped variable star with the earth in the center.... Sometimes I think that such a project could be simpler than trying to make a magazine exactly as we think it should be.)

Some members of the editorial board completed their terms—we must thank them for their contributions—and others joined the board. Our editorial board is growing with capable people from academia and industry, while it keeps the international balance between the US, Europe,

and the Far East.

A magazine results from the combined efforts of authors, editors, staff, and readers. The first three groups of people depend on the last—you, the readers—to continue to support the magazine by providing proposals, comments, and suggestions. As in the past, we promise to always consider these comments.

What changed in the magazine during 1991? Costs were reduced (credit to our staff), allowing us to bring you the most pages possible in spite of inflation and general economy problems.

We have also devoted efforts to reducing the amount of time spent in reviewing manuscripts. The shorter review cycle serves the needs of authors (their work reaches a wide audience in a shorter time, or at least they know their fate sooner) and readers (they receive updated information quickly). The theme issues in 1991 brought coordinated sets of articles on some of the hottest areas in microelectronics and microsystems. This will continue in 1992 and 1993. (See p. 41 for clues.)

The current issue is a "general" one, so you find an assortment of articles that address various aspects of high-performance computing. The first, "The Scalable Coherent Interface and Related Standards Projects" by Dave Gustavson, provides an insight into an IEEE project (P1596-SCI). SCI proposes new solutions for the communication bottleneck intrinsic in bus-based multiprocessor architectures. Instead of wider and faster buses (which also means more expensive and power-hungry buses), we can switch to nonbused interconnection schemes and still keep the configurability we generally find in bused systems. SCI is actually a set of projects addressing the many aspects of an interconnection structure, from the lowest level (the physical

communication channel) to higher protocol layers dealing with cache coherency. It ensures tight coordination with other standards projects.

Next Daniel Mann's "Unix and the Am29000 Microprocessor" article shows how the hardware features of a RISC processor can be exploited to optimize its performance for a given operating system.

With the third article we switch to completely different architectures. "Hardware Requirements for Neural Network Pattern Classifiers: A Case Study and Implementation" by Boser et al. describes in detail an ASINC, or application-specific integrated neural circuit. This is so new an acronym that it does not yet appear in Carl Warren's glossary (see Micro Standards beginning on p. 69).

Efficient and affordable handwritten character recognition can open the way to a variety of new industrial applications. I am convinced that we will see such features within the sensor itself in a few years (the same way keyboards now have embedded controllers that send ASCII characters directly to the host microprocessor). There is a good chance that such intelligent sensors will use the neural approach, so take a close look at this article to get an idea of what will be on the market in the near future.

A common denominator for these three articles is that each of them achieves high performance, thanks to some specific silicon basis (protocol chips, RISC, neural hardware). The next article, "Experimentation with Hypercube Database Engines" by Frieder et al., describes efficient algorithms for handling distributed shared data in a specific architecture.

Even the best hardware and the best software can provide good results only if they can work correctly. In a modular system, this means each module must comply with the specification of the common interface (the bus). The last article in this issue, "Conformance Testing of VMEbus and Multibus II

Products" by Adams et al., describes a technique used to verify compliance of boards to a given bus specification.

faute hel com

In the mailbag

(LK: liked; DLK: disliked LTS: like to see)

In this mailbag the total number of readers who did not like split articles amounts to nine. Their comments are not individually reported here; I think they already received their answer in December. Thank you all for the advice.—D.D.C.

February 1991

LK: Hardware design; LTS: network systems.—S.T., Moscow

April 1991

LK: Optical architecture: excellent article, though I think there were some errors in certain figures and not enough clarity in parts of the text. (Please be more precise, so we can inform the author and ask for corrections if required.—D.D.C.) LTS: RISC architectures.—J.C.A.A., Tlalnepantla, Mexico

LK: Software and hardware computer science and its electronics; LTS: ... U.P.S. units for computers.—M.M.M., Alexandria, Egypt

LK: Letters, Micro News, Micro View, and other chapters; LTS: a chapter [identifying] computer idioms.—M.G., Isfahan, Iran (An acronym dictionary appears in this issue.—D.D.C.)

June 1991

LK: Richard Stern, Book Review—T.S., Tromsø, Norway

LK: On The Edge; it has focused very well on the problem. Moreover, author used simple language to describe it ... very important ... to teach something. I'd like to see more articles like this.—C.G., Verona, Italy

(These compliments are for Carl Warren and James Gafford, who were responsible for this column.—D.D.C.)

LK: The whole issue; DLK: "Light at the end of the chunnel." It has nothing to do with the hot chips [themel. (You are correct: Departments are not necessarily linked with special themes.—D.D.C.) LTS: Hot Chips III—T.P., Warsaw (You will get it in the April issue.—D.D.C.)

LK: The overview of the BTRON/286 specs.—E.D., Aalen, Germany

LK: Enjoyed reading [about] iWarp and Datawave. How much does the iWarp and ITT Datawave chip cost? (The task of the designer—who writes the articles—is to minimize the manufacturing cost, but they usually have little control on final pricing. It is better to ask a commercial representative.—D.D.C.) DLK: the guest editors' introduction of what is superscalar and pipelining; LTS: How superscalar and pipelining come into play. Do all chips have one or could they have both? This was not clear.—A.R.F., San Ramon, CA (Almost all microprocessors announced in the last five years are also superscalar.—M.D.H./D.A.W., guest editors.)

DLK: Rate monotonic scheduling, too little tutorial information.—P.F., Adliswil, Switzerland

LK: iWarp: A 100 MOPS....—S.R.E., Saudi Arabia

August 1991

LK: Micro Law; I never miss it.—T.D.L., Cupertino, CA

LTS: More practical software reports for science and engineering, Unix.—F.M.R., Munich



Micro Law

Richard H. Stern

Oblon, Spivak,
McClelland,
Maier & Neustadt, P.C.

1755 Jefferson Davis
Highway

Suite 400

Arlington, VA 22202

Engineers can be disqualified, too.

During the heyday of the mergers-and-acquisitions frenzy, the disqualification ploy became a common litigation tactic. The first thing a party would do in a lawsuit would be to file a motion that the other side's lawyer should be disqualified from representing his or her client, because at some time or another counsel had performed some kind of legal representation of the adversary party who made the motion.

The rationale for filing such motions, and for the court when it went along with the move, was to prevent the appearance of impropriety. Never mind that the earlier legal representation had nothing to do with the present case. The public's great confidence in the legal profession might be impaired if it even appeared that clients might not be able to rely on the confidentiality of their relationship with their attorneys. How could clients feel safe hiring lawyers and telling them their secrets if the very same secrets would later be used against them to help an adversary?

After a while, the courts became as cynical about these motions as those who filed them and recognized them as another ploy to put the other party at a tactical disadvantage. They became skeptical about arguments based on the body blow to our social order wrought by the appearance of impropriety and began to limit disqualification to situations where a party's adversary would actually get the advantage of relevant business secrets disclosed in confidence in an earlier case. Such things as knowing the structure of the client's hierarchy of values, the client's aversion to or enthusiasm for risk, how the client thinks about things in general, or the client's negotiating strategies, all went out the window as bases for disqualification.

The rule now seems to be pretty close to this:

The lawyer must really know (or be likely to know) where the relevant bodies are buried. For example, the court may disqualify counsel if the lawyer

- wrote the patent application on the invention that the lawyer now wants to assert (on behalf of an adversary of the inventor) is invalid;
- is going to be a witness, because he or she wrote the patent application that supposedly was used to defraud the patent office;
- represented the patent owner in another case involving a related patent, where counsel learned about the weaknesses of the client's claim on this technology; or
- represented one party to a license in working out and drafting the agreement, where parties are now in a dispute over what the agreement was intended to do (what it means), and the lawyer wants to represent the other party in the dispute (or maybe now the lawyer plans to claim that the original license agreement is legally invalid).

While things are pretty well sorted out on that front, a new frontier is opening up, which may be of greater concern to electrical engineers and computer science professionals. It now appears that they, too, can be disqualified.

The Balde case. In a decision handed down last summer, *Wang Laboratories, Inc. v. Toshiba Corp.*,¹ a federal court in Alexandria, Virginia, (see box) disqualified a computer consultant from representing NEC (a codefendant of Toshiba) because he had previously given Wang a preliminary opinion that its patent was invalid. In November 1990, Wang's attorney had telephoned John Balde, a computer consultant. He asked

Balde if he was familiar with SIMM (single in-line memory module) technology, and Balde said he was. The parties differ over what happened next. According to Wang, the lawyer retained Balde and agreed to pay him for his time. According to Balde, he was not retained, and he told the lawyer he would have to determine whether Wang's SIMM patents were valid before he would enter any agreement with the company.

Wang's lawyer then sent Balde a letter (dated November 14) transmitting various documents: the SIMM patents, some prior art publications, some materials concerning patent infringement, and a long memorandum written by Wang's attorney about the history of the prosecution of the patents before the patent office. He asked Balde to review the material, "so that we can discuss how best to explain the advantages to a computer designer of using" SIMM, and he suggested they meet after Balde reviewed the material.

The next day—watch out for this one, readers—Wang's lawyer sent a second letter. Unlike the first one, it was prominently labeled, "Confidential Attorney-Work Product." This letter (dated November 15) contained an outline of potential legal defenses against Wang's suit, as Wang's lawyer perceived them, and asked Balde to provide his opinion on various issues. It also said that this material "will assist your review of the material included in my November 14, 1990, letter."

Several telephone conferences followed. The lawyer later said that in them he disclosed Wang's confidential information to Balde, and that he made the confidentiality clear to Balde. Balde did not deny this, but said that he made no use of the material in the November 15 letter. He considered the letter premature because it asked him to give opinions on specific litigation issues. But he did not want to give any opinions on the issues until he had made up his mind about the validity of the SIMM patents. If they were not valid,

Alexandria's "rocket docket"

Last summer, the US District Court for the Eastern District of Virginia awarded Wang \$3.3 million in damages against Toshiba and NEC for patent infringement. The case is of special interest because it comes from the "rocket docket" of Alexandria, Virginia. This court is becoming a forum of choice for patent infringement litigation and other complicated cases that usually drag on for many years, because of its firm policy that all cases *must* be tried within six months after joining issue. Presumably, one can expect the same kind of rulings about experts in other cases brought in that court.

he did not want to become involved with Wang. Balde felt that Wang's lawyer was jumping the gun on enlisting Balde in Wang's camp before he felt ready to sign up. Nevertheless, as the judge pointed out in his opinion, Balde did not write back to Wang's lawyer saying any of this.

Balde studied the matter and concluded that the SIMM patents were invalid. On December 10 he telephoned Wang's lawyer, told him his conclusion, and said that he therefore did not want to act as a consultant for Wang in the case. The lawyer asked Balde for a report, which Balde sent two days later. In his cover letter for the report, Balde wrote, "As you know, I have read ... the Work-Product information on the two Wang SIMM patents," and he thanked Wang for offering to pay his \$1,500 invoice for time spent on the task. Subsequently, NEC retained Balde as its technical expert in the case. Wang then moved to disqualify Balde.

Court's decision. The court found little precedent about disqualification of experts, but held that it had an in-

herent power to disqualify them in appropriate cases. This power exists to help the court fulfill its judicial duty to protect the integrity of the adversary legal process and promote public confidence in the fairness and integrity of the legal process.

The court found two issues: 1) Was it reasonable for Wang to think it had a confidential relationship with Balde? 2) Did Wang disclose confidential information to Balde? If and only if both answers are affirmative, Balde should be disqualified. It also recognized that lawyers might seek to disable potentially troublesome experts merely by retaining them without using them, and said that it would not countenance that ploy. (Here, Balde had been NEC's consultant in the past.) On the other hand, consultants who do not want to be bound by a duty of confidentiality should make it clear that they do not want to be retained until they make up their minds, and in the meantime they should not accept confidential disclosures.

In this case, the November 15 letter carried the day for Wang. The letter was captioned "confidential attorney-work product," and an examination of the enclosures confirms that description. The court said, "No experienced litigator would freely disclose these materials to opposing counsel." Balde's silence in the face of the November 15 letter established that Wang's lawyer was reasonable in thinking that a confidential relationship existed, and in fact that letter transmitted confidential information to Balde.

Advice. Neither Wang nor Balde had "acted inappropriately," the court found. Even so, it decided to offer some free advice for others "to avoid a repetition of these unfortunate circumstances." First, lawyers should make it clear and confirm in writing that a confidential relationship will be created. Preferably, this should be specifically explained in a letter, along with confirmation of payment terms and conditions. These elements were missing in

this case. Nevertheless, the court disqualified Balde.

Next, consultants should take care to avoid creating confusion about their position. Doubts about wanting to be retained should be unequivocally expressed. Here, the court said, "given his stated [read that as 'alleged'] concerns" about patent invalidity, Balde needed no more than the identity of the parties and the patent numbers. (The last is a bit of judicial overkill. Since the patents are public documents, their text is no secret. It would be an inconvenience to make Balde get his own copies.) He should have declined to accept anything more, the court said.

Counsel should ask the consultant whether his past employment creates any problem. Since NEC apparently did not make this inquiry of Balde before retaining him for the case against Wang, the court felt that NEC had only itself to thank for its problem of having no expert on the eve of trial. (Presumably, Wang, too, should have asked Balde whether his past work for NEC gave him any of NEC's secrets, which could make it improper for him to be Wang's consultant.)

Also, NEC should have promptly advised Wang and discussed the matter "thoroughly in an effort to resolve the dispute before it is raised in court." (To which I say, lots of luck. Why would Wang agree to roll over on anything? Excess of gentlemanliness? When was the last time you heard of noblesse oblige being a significant factor in how litigators behave?)

To be sure, the court said, experts "are not advocates; they are sources of information and opinions in technical [matters] ... Yet, when experts are retained in connection with litigation, they must operate within the constraints of, and consistent with, the adversary process."

I am not going to express any opinion on whether Wang's lawyer sandbagged, hoodwinked, or otherwise mistreated Balde and NEC, nor on whether Balde got a deserved come-

uppance. The court apparently felt that none of that happened, which is, of course, quite good enough for me. But the lawyer certainly outmaneuvered the EE in this case.

Protecting yourself. So where does all of this leave other EEs? What do you do to "avoid a repetition of these unfortunate circumstances"? Or, exercising 20-20 hindsight, what should Balde have done? What would the *CYA Manual for Electronic Engineer and Computer Science Would-be Expert Witnesses* have prescribed, if anything? Remember, Balde was not trying to scare Wang away. He didn't know whether NEC would want to retain him in this case, and he had to pay the rent every month. He also did not know, presumably, how he felt about the SIMM patents and Wang's case.

**... consultants ...
could be
"Balde-ized."**

Ideally, Balde would have had his own lawyer, whom he would have consulted immediately after the November 14 telephone call. He should have turned the November 14 and 15 letters over to a lawyer, unread and preferably unopened. Balde's lawyer would have responded for him or advised him how to respond. He would have matched Wang's lawyer's efforts with the same thing in reverse, something like the following:

*F, U, & D, Counselors-at-Law
November 16, 1990*

Dear Sir:

My client, Double-E Balde, has turned over to me your letters of November 14 and 15 and enclosures, which he has not read. My client wishes you to be advised that he is not yet fully

prepared, at this time, to enter into a relationship of confidentiality with regard to the subject matter of Wang v. Toshiba and NEC. In all fairness to Wang, my client feels that he should first make his own preliminary evaluation of materials already of public record, or otherwise not secret or confidential, in order to satisfy himself that he can appropriately consult with, or act as an expert for, Wang in this matter. I am sure you will appreciate that it would not be in Wang's or his interests to have him prematurely enter into a confidential relationship with Wang, in the event that it later appears he would be obliged to testify as to opinions inconsistent with Wang's position in this matter.

We will get back to you as soon as Mr. Balde has finished the foregoing preliminary evaluation. In the interim, I will retain the documents, unless you wish me to return them to you at this time.

*Sincerely yours,
F, U, & D*

Of course, you may feel it is not feasible to run a consulting business this way. What then? You consultants who don't want to run your business through lawyers still must do something, or you could be "Balde-ized." At the very least, you should send a letter, such as the following, that negates a confidential client relationship like the one set up for Balde by the lawyer's letter.

*Bit Bucket Consulting Services, Ltd.
November 16, 1990*

Dear Counselor:

Thank you for your letters of November 14 and 15, 1990, regarding the pending patent infringement suit between Wang and Toshiba/NEC regarding SIMM technology. I am responding to both letters at the same time, because your second letter arrived before I could

continued on p. 72

Software Report



David K. Kahaner

US Office of Naval

Research, Far East

kahaner@xroads.cc.

u-tokyo.ac.jp

R&D in Japan

These two reports and two announcements reflect some of the ongoing research in this country.

Forecast for 2010

Japan's Economic Planning Agency enlisted the assistance of a group of 10 experts to assess the country's position in technology and its direction for the next 20 years. After listing 101 technological items, the study group developed a questionnaire concerning them, combined the responses, and wrote a lengthy report and summary (in Japanese). The summary, released in July 1991, proved fascinating in the detailed views of the group members but produced some problems for readers. For example, the study may not have much statistical validity, given that one or at most two experts assessed the individual items. In addition, parts of the textual material were awkwardly phrased.

But to me, the most interesting portions of the report appear in its tables. Table 1 lists selected items from the report's tables.

Micromachines

Under Japan's National Research and Development Program (popularly known as the Large-Scale Project), industry, government, and academic circles cooperate on research and development of innovative, advanced, large-scale industrial technologies deemed important and urgent for the national economy. Since the program began in 1955, 29 projects have been launched, eight of which continue today. MITI (Japan's Ministry of International Trade and Industry) proposed an R&D program called Micromachine Technology to begin in fiscal year 1991.

The New Energy and Industrial Technology

Development Organization (NEDO) under the authority of the Agency of Industrial Science and Technology will conduct the Micromachine Technology project. NEDO plans to establish the technologies necessary for the realization of micromachines.

Elemental technologies. The first step of the program (covering the first four to five years) emphasizes the establishment of basic technologies of elemental components for micromachines and targets four major R&D items. The first major item in this phase will establish technologies for material processing and control of dynamic mechanisms by developing the elements of a micromachine. The elements are actuators (thermal-, electrical-, or magnetic-induced deformation, electrostatic, hydraulic) and sensors (electromagnetic, chemical).

A second item calls for the development of technologies that transform energy from external sources, micro internal batteries, or micro power generators. A third will investigate micromachine communication with the exterior world and conduct theoretical research and associated software development of remote control and coordinated distributed control.

A final major item concerns the measurement of accuracy and movement of the micromachines for evaluation of the results of the R&D program.

R&D description. Though micromachines are small, they are complex systems with advanced functions. Some of the areas that need to be researched include the basic theories that underpin miniaturization methods, structural analysis, materials, and component technologies of microscopic processing and assembly. Other areas include the techniques for producing microscopic sensors and control circuits, and the system technology required to perform microscopic motion

Table 1. Identified technologies and application year.

Technology	Year
Information electronics	
Biosensor	2000
Superparallel computer	2010
Terabit optoelectronic file	2010
Superintelligent chip	2010
Terabit optocommunication device	2010
Automatic translation systems	2020
Virtual reality system	2020
Self-replicating database system	2020
Neurocomputer	2030
Terabit memory	2030
Self-replicating chip	2050
New materials	
Ceramics gas turbine	2000
Magnetic materials	2010
Hydrogen occlusion alloy	2010
Optical IC	2010
Superlattice devices	2010
Nonlinear optoelectronics	2020
Superconductors	2030
Automation	
Micromachines	2010
Concurrent engineering	2010
Intelligent CAD	2020
Communication	
TV conference system	1994
TV telephone	1994
Optoelectronic LAN	1995
Broadband ISDN switches	1995
HDTV	1995

and operation. Table 2 on the next page lists some of the main topics envisioned for micromachine technology.

The technology. Microscopic machines or instruments with advanced functions can perform minute tasks or work in extremely narrow spaces. Their small size allows them to be applied to a wide variety of areas, including medicine, biotechnology, and industry.

Recently, silicon micromachining technology, in which micrometer mechanical structures are formed on silicon wafers, opened up this field. The technology emerged from etching, deposition, and other lithographic techniques for microprocessing silicon de-

veloped in the 1970s and enabled the production of cantilevers, diaphragms, and other simple mechanical components. These products are now used widely as pressure sensors or are beginning to be commercialized as acceleration and flow sensors. Moreover, recent advances in semiconductor microprocessing and ultraprecision processing technology allow the creation of mechanical parts far smaller than anything previously developed.

Researchers only recently began to investigate micromachine technology, however, and need to overcome many technological barriers before such machines can be developed. Some barriers

are those related to friction, durability, strength, materials, and power sources and supplies. Researchers also must develop a number of other technologies, including ways to design, process, assemble, and control micromachines.

Micromachines today. We can approach micromachine development in two ways. One approach uses technology in the field of mechanical engineering, which makes existing mechanisms even smaller; the other uses Micro Electro Mechanical Systems (MEMS) technology, which uses IC production technology. Many researchers have proposed or built prototypes of various microactuators and microstructures that could serve as the component technologies for micromachines.

In the United States, schools like MIT, Stanford, and the universities of Wisconsin-Madison, Michigan, and California, Berkeley continue to research surface micromachining technology and LIGA (Lithografie, Galvanofomung, Abformung) process technology in their silicon and LIGA process research centers.

MIT researchers produced a 100- μ m-diameter polysilicon micromotor rotating at 15,000 rpm and analyzed its movement. Researchers created the motor using the same process used for IC production. Wisconsin-Madison researchers produced 3D structures containing gears with 55- μ m inside diameters. Researchers at Berkeley produced a 120- μ m-diameter electrostatic motor and verified that it does rotate. They can also measure friction coefficients, one of the most difficult problems in micromachine technology, using a micro electrostatic linear actuator.

The US National Science Foundation supports these efforts. In 1988 NSF distributed its micromachine research budget among eight universities and in 1989 provided funding to 11 universities.

Europe also supports several micromachining facilities, including Germany's FraunhoferInstitut, Technische

Table 2. Main topics of micromachine research.

Technologies	Description
Microscopic mechanical device	R&D on structures, materials, machining techniques, integration techniques, and power supplies for microscopic mechanisms and functional components required for micromachines Development of technology to enable the production of various mechanical devices
Microscopic sensors, control circuits, and other techniques for miniaturized electronic devices	R&D in the technologies needed to produce extremely miniaturized electronic devices such as microscopic sensors and control circuits used in micromachines
Control and operation	R&D in motion control and operation technologies for microscopic mechanisms
Measurement and evaluation	Basic research into measurement methods, evaluation methods, and microscopic measurement technology as they relate to various component devices
Support	Basic research into support technologies including lubrication techniques for microscopic parts, theoretical simulation, and CAD/CAM
System integration	R&D project extending 10 years (FY1991-FY2000) and costing approximately 25 billion yen

Universitat Berlin, Kernforschungszentrum Karlsruhe Institut für Mikrostrukturatechnik, and the Netherlands university of Twente. Research concentrates mostly on sensors. The Fraunhofer Institut für Mikrostrukturatechnik produced prototypes of a vibration sensor with 32 cantilever mechanical resonators and a 1.5-mm × 1.25-mm cantilever thermal bimorphic microactuator. These facilities receive subsidies both from the European Economic Community and from their respective countries.

Japan pursues numerous creative studies related to micromachine technology. For example, the University of

Tokyo developed prototypes of a micro Stirling engine with high thermal efficiency as a microactuator, Tohoku University produced a microvalve using silicon. NTT Applied Electronics Laboratories produced a 500-nm × 500-nm, active integrated optical microencoder with 0.01-μm resolution.

Research at many universities, national research institutes, and private companies continues to produce prototypes. These include electrostatic linear actuators, micropressure sensors, micro IS-FET (ion-sensitive field-effect transistor) sensors, micromanipulators using piezoelectric-impact drive systems, micro active catheters using shape

memory alloy (SMA), and more.

These technologies, which mainly use semiconductor production techniques, represent only a small part of micromachine production technology. Research in this field is still in its infancy, and many problems remain to be solved. Some of the hurdles to be overcome include the development of production and process technologies geared specifically toward micromachines and solving questions related to friction, durability, strength, materials, power supplies, and control.

Application of results. Since micromachine technology will have a variety of applications, the program will focus on common component technologies. Once developed these areas probably will result eventually in applications like the following.

Industrial micromachines. Industry faces the need to boost reliability and reduce maintenance costs for ever more-advanced and complex mechanical systems and equipment (power plants and airplane engines are two good examples). A tremendous need exists for technology that makes it possible to perform inspections and repairs in extremely tight spaces without having to dismantle the entire system or equipment in question, such as plant pipe systems and airplane engines.

Industrial micromachines will enable inspection and repair without requiring that plant equipment be dismantled. This capability will make it possible to perform early inspection and repair to minimize the extent of damage. We can therefore expect significant improvements in capacity utilization and maintenance costs for electric power plants and other facilities.

Medical micromachines. Today's medical procedures do not sufficiently alleviate the pain experienced during diagnosis and treatment. In addition, as the population ages, we will see a strong demand for advanced medical equipment that lessens the physical and mental stress inflicted on patients.

continued on p. 87



The Scalable Coherent Interface and Related Standards Projects

The Scalable Coherent Interface (IEEE P1596) provides bus services by transmitting packets on a collection of point-to-point unidirectional links. Its protocols support cache coherence in a distributed shared-memory multiprocessor model, with message passing, I/O, and LAN communication taking place over fiber optic or wire links. Several ongoing SCI-related projects apply the SCI technology to new areas or extend it to more difficult problems.

David B. Gustavson

*Stanford Linear Accelerator
Center*

The Scalable Coherent Interface (SCI) was developed by a number of bus designers and system architects who had come to understand the fundamental limits to bus technology during their work on Fastbus (IEEE 960) and Futurebus+ (IEEE 896.x). These modern buses push bus signaling technology to its limits and provide various architectural features that support the use of multiple processors.

These bus limits are rapidly becoming a serious problem as the demand for computing power continues to grow. The economic reality is that we can only meet this demand by using a large number of fast microprocessors. Buses, however, are inherently a bottleneck (only one transfer at a time), and their signaling speed is limited by the imperfect transmission lines that result from bus-style connections.

Therefore, buses can't support a large number of processors, especially not fast ones. While we can extend their useful life a bit by cleverness and brute force, the potential gains are relatively small and the costs become very high. For example, doubling the width of a bus does not double its speed because there are fixed overheads associated with arbitration and addressing. Lengthening block transfers to reduce the effect of these overheads is of little use once the blocks exceed the size of cache lines.

We can increase signaling speeds by shortening the bus, but that makes it less useful. Reducing the signal voltage helps, but eventually this solution experiences noise problems. Using multiple buses to achieve more than one transfer at a time results in a complex (expensive) bus-bridge mechanism to maintain cache consistency (coherence) in shared-memory systems that use bus-snooping technology.

Paul Sweazey (Futurebus cache-coherence task group coordinator) started the Superbus Study Group in November of 1987 to see if there were potential solutions to these problems. In July 1988 the outline of the solutions had become clear, and the P1596 SCI working group replaced the study group. The work was essentially completed by January 1991, when specification draft D1.00 went out for ballot. Since then, it has been undergoing minor improvements, polishing, and debugging of the specification C code. (Most of the SCI specification is executable, to reduce ambiguity, simplify testing, and enable accurate simulation.)

The resulting draft D2.00¹ recirculated to the balloting body in December. Since the draft passed with 92 percent affirmative, and all but one objection has been resolved, (we refused to change the C code to Pascal), final approval by the IEEE Standards Board seems probable in March 1992, unless new objections arise.

SCI goals

The SCI design goals include

- *Scalability*, so that the same mechanisms can be used in high-volume, single-processor (or few-processor) systems such as one might find in desktop machines, as well as in large, highly parallel multiprocessors (next-generation supercomputers);
- *Coherence*, to support the efficient use of cache memories in the most general and easiest-to-use multiproces-

sor model, distributed shared memory; and

- An *interface*, a standardized open communication architecture that allows products from multiple vendors to be incorporated into one system and interoperate smoothly.

Scalability keeps costs down, not only through increased volume of production but through the simplicity of having to learn only one new paradigm—one that will work over several generations of machines. (See box.)

SCI applications

SCI uses point-to-point links to achieve very high speed communication. For the highest performance over short distances (typically within a cabinet), 16-bit-wide links run at 1,000 Mbytes/s. For I/O applications within a room, serial coaxial cable links run at 1,000 Mbps. For I/O over campus distances of a few kilometers, optical fibers can carry the same serial bit stream. See Figure A.

SCI's scalable architecture allows the same protocols

to cover the range from internal communication within a multiprocessing supercomputer to local area network applications. LAN communications look like moving data from one address to another in memory, a very simple software model. However, wide-area networks need hierarchical addressing instead of SCI's flat 64-bit address model, so the usual software protocol translations are necessary.

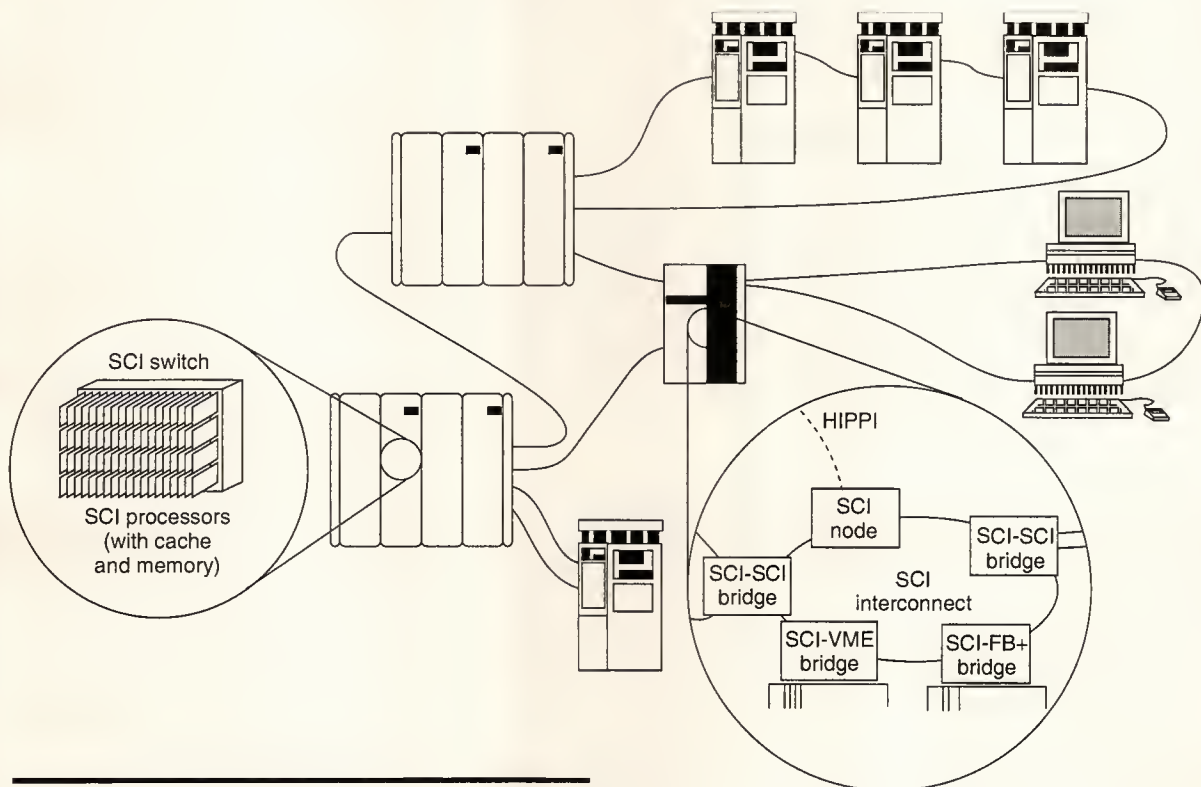


Figure A. A typical SCI application.

A standard SCI module defines signals, connector, and power for operation at a link speed of 1,000 Mbytes/s. A standard cable and connector defines the use of these signals for applications in which the module form factor is not appropriate, over short distances (meters). A standard fiber-optic serial interface solves interface problems over longer distances (kilometers) at a link speed of 1,000 Mbits per second. The same bit stream may be sent over coaxial cable at lower cost for medium distances (tens of meters). We plan to standardize other speeds and signaling in the future as appropriate.

Solving the bus signaling and bottleneck problems.

SCI needed two fundamental changes from the way a bus transmits information. First, to make signaling speed independent of the size of the system, interfaces don't wait for each signal to propagate (that is, a bus cycle) before sending the next signal. Each communication takes place by sending packets that include an address, command, and data as needed. While the propagation velocity of a packet is still limited by the speed of light, its rate of information transfer is not. As technology advances, the transfer rate can increase indefinitely. Fortunately, computer scientists know techniques that can compensate for the bad effects of delay (latency), but little can be done to compensate for a transfer rate (bandwidth) that is too small. (See Transaction phases box.)

Secondly, SCI uses multiple signal paths (links) so that multiple independent transfers can take place concurrently. For high performance, designers can use separate links for each processor, memory, or I/O device.

We further refined the SCI link design by applying lessons learned in practical multiprocessor systems. The link signals are differential because differential signals produce the least system ground noise and are least sensitive to noise from other sources. The links are unidirectional because bidirectional links create noise when the drivers are turned off or on to reverse the direction of the link, and because turn-around delays increase with cable length, a scalability problem. The links are fast and narrow because we expect pins will always be relatively expensive.

SCI does not use reverse-direction flow control signals because such mechanisms make the amount of buffer storage needed in the interfaces dependent on cable length. To reach high speeds, we use low-voltage differential signals. SCI initially uses 16-bit-wide, ECL-compatible signals because of the industry experience with and support for that standard. Future links will probably use even lower voltages that are chosen for compatibility with VLSI CMOS or GaAs circuitry.

Thus each SCI interface (node) has (at least) two links, one incoming and one outgoing. The links run continuously, sending idle symbols when no packets are being transmitted, so that the receiver can remain perfectly synchronized at all times, ready for action. SCI packets do not need the prologue that is essential for Ethernet or similar networks.

In a high-performance SCI system, a vendor-dependent switch accepts packets from nodes and routes them to the appropriate other nodes as specified by the address in the packet. SCI does not specify the details of such a switch, because many cost/performance trade-offs exist.

Lowest cost SCI systems, such as desktop systems, typically will use a ring connection instead of a switch, connecting one node's output link to its neighbor's input. The decision to support rings made the interface circuit more complex because a node may receive a packet intended for some other node and have to pass it along. That process requires some buffer memory and some address recognition logic. However, the result is that one interface definition works over a very wide range of applications, increasing the production volume and lowering the costs for everyone. (See Node interface structure.)

Point-to-point signaling is much easier than bus signaling. This, in combination with SCI's low-current, single-voltage (48V) power dis-

Transaction phases

As seen in Figure B, SCI transactions handshake on a packet basis rather than on a bus cycle basis. A request packet contains all the necessary address, command, and possibly data needed to initiate a transaction. The packet is either accepted and stored in the responder's queues or (if there is insufficient space) is discarded. An echo tells the requester whether it can discard its send packet or must retransmit later because it was not accepted. A similar handshake occurs on the response.

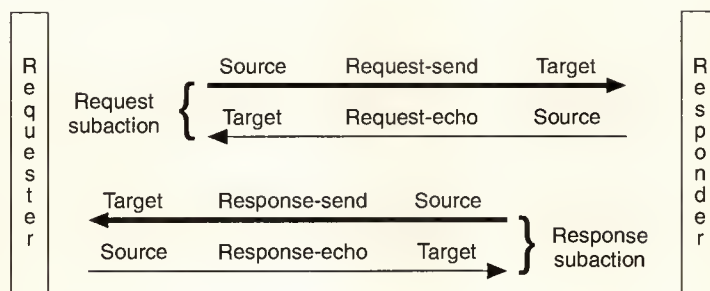


Figure B. Transactions have requests and response subactions.

Node interface structure

Data arriving on the incoming link shown in Figure C have to be resynchronized to the node's own clock. The receiver's elastic buffer circuitry handles this step. The rest of the node circuitry is synchronous, which greatly simplifies the design of the ultrafast FIFOs and logic.

If a node receives a packet intended for some other node while it is transmitting a packet of its own, part or all of the incoming packet moves to the bypass FIFO.

When no packet is being received, idle symbols keep the receiver synchronized and carry information about the priorities of other nodes. They also carry Go bits, which act like tokens and help ensure fair use of the links. Fair use of part of the bandwidth is important in avoiding starvation or deadlock.

An SCI node maintains dual queues to keep responses independent of requests. Without dual queues, excessive requests could prevent the sending of responses, resulting in deadlock.

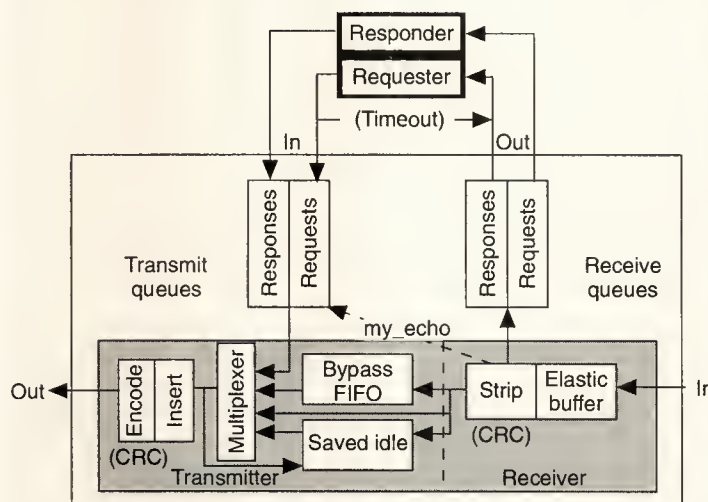


Figure C. Block diagram of a typical SCI node.

tribution system, should make ring-connected backplanes very inexpensive. A few printed-circuit-board layers will generally be enough.

Intermediate-level SCI systems may use a combination of switch and ring, using paired SCI interfaces as a simple bridge that connects two rings. By combining many rings, designers can build a distributed switch fabric. Active research is in progress to discover optimal configurations.^{2,3} Scott and Goodman² show that when pipelining is permitted, as is the case for SCI, we gain performance rapidly relative to synchronous systems by increasing the dimensionality of the interconnect. The resulting performance is much higher than for packet-synchronous systems.

SCI defines efficient packet-based protocols that provide the kinds of services we expect from a computer bus. The main differences seen by the user relate to the separation of the request for service from the response. A simple bus waits during a memory read access time, until it gets the data. No other parties can use the bus during that wait. This approach makes it conceptually easy to handle error conditions or to perform complex mutual-exclusion operations (like read-modify-write). See the Transaction formats box.

More sophisticated buses split the operation into a request and a response phase, just as SCI does. Then the interface must keep track of pending requests to match the responses to them, and a different style of mutual exclusion becomes necessary. Engineers who have already used split-response

Transaction formats

SCI packets have a 16-byte header that contains address, command, transaction identifier, and (in a response) status information. All packets that may need storage space in queues are multiples of 16 bytes, to simplify storage management at very high speeds. The echo packet is an 8-byte subset of the header (not shown in Figure D); it is never stored in queues. A few transactions need an extended header (not shown either), which adds another 16 bytes.

	Request		Response	
readxx*	Header		Header	0,16,64,256
writexx*	Header	16,64,256	Header	
movexx*	Header	0,16,64,256		
locks	Header	16	Header	16

* xx represents one of the allowed data block lengths (number of data bytes, on the right after the header).

Figure D. SCI packets.

buses will find SCI to be clean and simple; those who haven't may face a learning curve.

Solving the cache coherence problem. Cache memories are important for keeping processors running at full speed, but they introduce some system management complexity. The various caches often contain duplicate copies of data that must be kept consistent with one another, the cache coherence problem. Say that two processors read the same data (perhaps a synchronization flag) from memory and cache it (perhaps on chip), and then one changes the data (perhaps to free a resource that the other is waiting for). Somehow the other processor must discover that its cached copy is invalid so that it can be updated with current information.

Designers have maintained cache consistency, or coherence, in small bused systems by taking advantage of the bus bottleneck—every cache controller observes every transaction in the system, “snooping” to catch transactions that might invalidate cached data. That approach can be scaled up to a few buses by making the bridges rather sophisticated, but it does not work in highly parallel systems.

A cache coherence scheme that scales to large systems requires a directory that keeps track of which data are being

used by which caches, so that the appropriate caches can be updated as necessary. Earlier schemes used a directory but kept it in memory. Instead, SCI maintains it as a distributed, doubly linked list of caches, with the head pointer at a memory controller and the link pointers stored in the cache controllers. With this approach we know that the correct amount of storage is always available for the directory structure, no matter how many caches are sharing copies of a particular line of data. This approach also spreads the maintenance traffic across the system, rather than concentrating it at the memory. Even though these linked-list structures are shared, we designed them to be updated concurrently by multiple processors without any semaphores or lock variables. The protocols use indivisible compare-and-swap transactions instead. (See the Distributed cache tags box.)

We can avoid the cache coherence problem if memory is not shared. Most of the early multiprocessors, which rely on message passing and explicit interprocessor communication, use this approach. However, it is often difficult to move computer programs from single-processor machines to message-passing multiprocessors. Note that shared-memory machines can easily pass messages, so they are more versatile. The

Distributed cache tags

SCI is a distributed system with many links carrying data independently and concurrently. Only directory-based cache coherence is practical in large high-performance systems of this kind. The directory keeps track of which caches are sharing each cache line of data, so that the right caches can be notified when their copy becomes invalid. Rather than centralizing the coherence directory (in RAM), SCI distributes it among those cache controllers that are sharing the data.

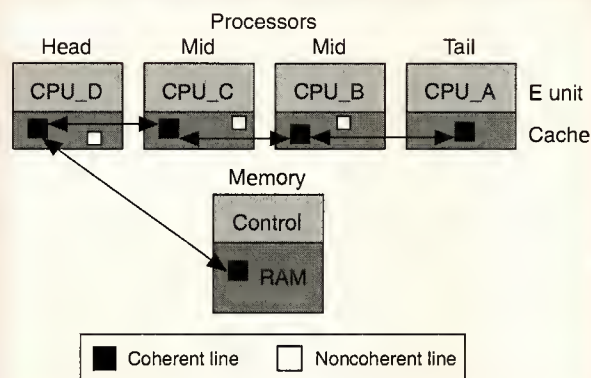


Figure E. A typical linked list for one cache line.

The SCI coherence directory always has storage available in exactly the amount needed, because the cache controllers sharing the given cache line provide it. The arrows in Figure E represent bidirectional pointers forming a doubly linked list. These pointers and a few status bits constitute the directory entry for this cache line.

The doubly linked list structure makes it possible for any cache to roll out a cache line to make room for new data, removing itself from the list by telling its neighbors to point to each other. This list maintenance traffic is distributed throughout the system, greatly reducing the concentration of traffic that is typical of centralized directory schemes.

The protocols support optional optimizations for important cases like pairwise sharing and serial resource allocation. Pairwise sharing lets two sharers pass data back and forth as needed without involving memory. Serial resource allocation (queue on lock bit) uses the list structure to pass the resource along without needless communication in the interconnect.

Note that the directory entry is a shared data structure that may be concurrently accessed by multiple processors. The SCI protocols rely on atomic compare-and-swap operations to ensure correctness without using semaphores or lock variables, which would be less efficient and introduce management complexity. For example, what do you do when a processor sets a lock variable and then dies?

Cache coherence

Until recently, microprocessor designers considered the use of a cache to be their own decision, based on cost/performance objectives, with little concern for the needs of multiprocessors. With just one processor, the existence of a cache usually affects its performance but not the correctness of its execution. A common exception occurs in the case of self-modifying code, where the processor executes instructions from the cache that are writing to memory in a futile effort to modify themselves.

However, in a system with two or more processors each of which has a cache, serious problems can result if the caches are not properly designed to work in harmony. For example, suppose two processors use a variable to keep track of which one is using the printer, and the software waits until no one is using it before starting a new print job. If two primitive processors read the same variable and each keeps it in its own cache, they each read only their cached copy, not any changes that the other processor may make to the variable. Designers must either arrange that such variables are never stored in caches or design the cache control so that each cache discovers when its copy of the variable is no longer correct, discards the old (invalid) value, and obtains a fresh copy.

Such mechanisms add cost and complexity, so it is not surprising that they were omitted in early caching (uni)processors. We call keeping all the cached copies of a data item consistent the cache coherence problem.

The cache controller can determine that its copy is invalid either by tracking every transaction in the system that might be capable of changing the data or by being notified

by a reliable source. Bused multiprocessors usually use snooping to maintain coherence. Each cache monitors the bus at all times, which keeps it fairly busy since it also has to be handling the memory requests from its processor. When the cache controller identifies another processor writing to the address of data it has cached, it can either pick up the new value on the fly ("snarfing") or mark the old one invalid and go to memory for a good copy when (and if) the processor needs it later. Say a cache can verify that it has the only good copy—because it changed the data—and it sees another processor trying to read the data from memory. It must then intervene to prevent the other cache from accepting a stale copy. Either the cache can supply the data itself or it can abort the transfer, update memory with the latest copy, and then let the transfer be retried.

The snooping mechanism relies on every controller being able to track every transaction. That's acceptable on a bus, because only one transaction can happen at a time, and each controller tracks it. With effort and care, snooping can even work in a system with several buses. But in a highly parallel system, like SCI, the cost of snooping is intolerable; too many transfers happen at the same time and in different places.

The usual solution to this problem is to maintain a directory that tracks which caches have copies of each cache line. (A cache line is the amount of data tracked as a single entity by the cache controller. SCI tracks 64-byte lines; earlier systems usually used smaller line sizes.)

The next decision is where to keep the directory and

continued on p. 16

goal, then, is to make the coherence protocols efficient and economical.

See the Cache coherence box for further amplification of the problem.

Multiprocessor issues. SCI designers placed a high priority on eliminating several architectural problems, such as deadlocks and livelocks, that have plagued past multiprocessors. Deadlocks can occur when two processors request the same two resources in the opposite order. Each processor may access one resource (blocking the other processor) then become blocked by failure to access the other resource. Livelock, or starvation, occurs when one processor repeatedly accesses a shared resource without letting another have its turn for an arbitrarily long time. Deadlock and livelock are examples of what we generically refer to as "forward progress" issues. To guarantee forward progress, each operation should result in some useful work done, so that the system will not

consume all its resources performing useless retries.

Experience with multiprocessor systems shows that they tend to synchronize themselves around problems like these. Even though the designer has estimated the probability of livelock to be extremely small, its effect on the system once it occurs is such that it tends to cause frequent repetitions.

Therefore, we designed the SCI protocol to eliminate dependencies that can cause deadlocks and to include a simple mechanism that assures fair allocation of resources to avoid livelocks. In a few cases, deadlock avoidance seems to make the protocol less efficient (until we consider the indirect consequences!). But in most cases choosing a clean protocol resulted in no penalty.

Of course, software that is outside SCI's scope can still create deadlocks. The SCI designers could only eliminate deadlocks from the underlying protocols.

SCI includes efficient mechanisms for supporting shared

resources, shared data structures, and mutual exclusion. Since the old standby read-modify-write is impractical in a parallel-processing environment, SCI designers took a fresh look at the problem and incorporated several mechanisms that will be helpful.

In a cache-coherent environment, the mechanism that guarantees exclusive use of a cache line by a writer can handle mutual exclusion. The processor can temporarily lock an exclusive cache line while it performs any operations it desires then release it to other readers or writers. However, coherence may not be available in all cases. For example, when accessing (through a bridge) a bus that does not support cache coherence, one may need to tell the bridge to perform a read-modify-write.

Therefore, SCI defines a set of lock primitives that experience shows to be useful for a variety of purposes: masked

swap, compare-and-swap, and fetch-and-add. Each of these primitives sends the command and necessary data to a destination device (perhaps a bridge or a memory controller). That device performs the operation indivisibly, returning the result to the requester. This mechanism works well through switches or other interconnects. (It even works well on buses.)

A particularly interesting use of the swap operations is in the maintenance of shared lists that hold information for an interrupt-servicing processor or commands for a DMA controller. If these lists are properly structured, multiple processors can add items to them while one processor takes items off, without any need for lock variables.

A clean way to handle interrupts is to associate a particular shared list with one priority of interrupt service and a specific bit in an interrupt-triggering control register. To request interrupt service, an I/O device or processor adds a work item

Cache coherence *(continued from p. 15)*

how to organize it. One way is to keep somewhere in memory a bit map for each cache line, setting a bit corresponding to a particular cache when that cache comes to the memory for the data. But in a large system, this would require too many bits. Using thousands of bits to account for the location of one cache line isn't acceptable. The next refinement might be to make a list in memory, keeping track of the identifiers of the caches that have taken copies. But while most cache lines are shared by perhaps only one or two caches, some might be shared by every cache in the system. Thus the storage for the worst case list becomes impossibly large. Designers have used a variety of clever compromises, such as allocating a small amount of list storage and then assuming the worst when it runs out. In that case, they assume that every cache in the system has a copy and must be notified when the data changes.

SCI features a very general and scalable directory structure. Memory controllers keep (and store in special memory) one node pointer and a few state bits for each cache line they have. That pointer points to the head of a list. The address of the data in a read transaction routes the read to the memory and tells the memory which cache line is desired. The read transaction includes the node identifier of the requester. The memory controller exchanges that identifier with its pointer for that cache line, returning the old pointer value to the read requester. If the memory has a current copy of the data, it returns the data to the requester too. Otherwise the memory informs the requester that the data is somewhere else, in another cache, and it can use the returned pointer to find it.

SCI cache controllers keep in their tag memory storage, for each line, the memory address of the line (like all cache

controllers have to do), a few state bits, and two pointers. The pointers form a doubly linked list of those nodes that have the particular line in their caches.

There are two particularly good properties of this scheme. First, it scales properly. No matter how many caches or how much memory or what the sharing behavior happens to be, exactly the right amount of storage is always available, namely two pointers per cached copy plus one pointer per cache line in memory.

Second, maintaining this list is a distributed process. Only one transaction touches the memory for each request. All the rest (following pointers, adding itself to the list, removing itself when its cache overflows and it needs to roll out a cache line to make more room) involve transactions among the distributed caches, which do not contribute to traffic congestion at the memory.

Though the bus-snooping mechanism may seem conceptually simpler, keep in mind that it bears a high price. It prevents us from having thousands of transactions proceeding at once. Furthermore, every cache in a snooping system must participate in every address cycle. Thus, every cache must be very fast so that participation does not slow the system too much, because the slowest one sets the speed for all.

In the SCI scheme the cache controller acknowledges a packet's arrival and checks it when it is convenient. If the controller is slow, it only slows the transactions in which it participates, not all transactions. As designers add new, higher performance, caches to a system, they get a proportionate performance improvement. This is another desirable kind of scaling behavior; designers don't have to discard old cache controllers to get the benefit of new ones.

Interrupts

Interrupts are simple in a single-processor system, because it is clear which processor should perform the requested service.

Some simple systems just use one signal line to trigger an interrupt. The interrupt causes the processor to save its state on the stack or in a duplicate set of registers, then execute software that asks all possible interrupt sources whether they need service—interrupt-driven polling.

More sophisticated systems allow the interrupting device to identify itself. Some kind of arbitration determines which source has the right to put its identifying vector on the bus during the interrupt-acknowledge cycle. The vector generally changes into an address at which the corresponding service code starts in memory. The arbitration mechanism varies from system to system. It may use a central method, so that individual signal lines connect to each interrupt source, or a distributed method, using a daisy chain or some other arbitration mechanism. Often a combination of polling and vectoring is used. For example, the service routine might poll a list of devices known to be connected to interrupt line 3.

These mechanisms are adequate for finding the source of an interrupt but do not deal at all with the problem of determining which of several processors should service it.

Systems with multiple processors require a more general mechanism. The most common solution provides a special control register associated with the interrupt system of each processor. To generate an interrupt, a device must become the bus master and write to the interrupt register located at the address corresponding to the processor to be interrupted. Becoming a bus master and executing a write may require additional hardware in the interrupting device. So, some multiprocessor systems (Fastbus, IEEE Std 960) also provide a hybrid capability that allows a simple device to assert a signal that causes a shared intermediary device to (possibly poll and then) perform the write for it.

It is tempting to design the interrupt registers to accept vector information directly. A device might write its as-

signed identifier into the register for use much like the vector used in simple interrupt systems. This temptation should be avoided, because there are hidden perils. For example, what happens if a second interrupt-write arrives soon after the first? Either it must be stored, implying a FIFO, or it has to be aborted and retried later. Any FIFO has a finite capacity, however, so under adverse conditions it might not be capable of holding any more vectors. But aborting the write and retrying also causes trouble, because it can lead to deadlocks. For example, suppose the interrupt service routine of one processor must send an interrupt to another, and the other processor has a service routine that has to interrupt the first. When both processors' FIFOs fill for some reason, deadlock ensues because neither can do anything except continue to retry sending the interrupt.

We can argue that these conditions are anomalous and designing the hardware to have large enough FIFOs will make the problem unlikely to occur. However, experience shows that real multiprocessor systems seem to seek out these trouble spots.

A clean solution puts the burden of allocating resources on the interrupt requester, so that it cannot ask for an interrupt unless it has the resources to ensure the interrupt request can be delivered. Suppose the requester allocates a block of memory to hold its vector and possibly other service information, links the block into a list of service requests, and then writes a pulse to the interrupt service register. Now, no possibility of being blocked exists, and no deadlock can occur.

We can design the service-request list so that any number of interrupt requesters can concurrently add their blocks to the list while one server removes blocks to service them. We will not need semaphores or lock variables, if the system supports indivisible swap and compare-and-swap transactions.⁴ These transaction types are extremely valuable in multiprocessor systems and deserve wide support by new processor architectures.

to the list describing what needs to be done, then sets the appropriate bit in the interrupt register. The item in the list is similar to the interrupt vector in single-processor systems. When the processor is ready to service that priority of interrupts, it takes work items off the list and services them.

This mechanism is clean because the service requester allocates all storage, so there is no danger of the server running out of storage when the work piles up (a possible cause of deadlock). The interrupt bit is simple to implement be-

cause it is only a latch, with no FIFO storage or critical timing implied. Setting the bit merely alerts the processor that the list should be checked; the processor can clear the bit as soon as it commits to look at the list. (See the Interrupts box for more information.)

A common use of mutual exclusion is to grant sequential use of one resource to a series of requesters. This process can generate a large amount of useless interconnect traffic as processors keep updating their cached copies of the shared ex-

clusion variable. SCI defines a queue-on-lock-bit, or QOLB, mechanism that uses the linked lists of the cache coherence system to pass the resource efficiently from one processor to the next.

Statistics on memory sharing are controversial, because few relevant machines exist and because the usage patterns on any real machine will evolve to optimize performance on that architecture. However, most researchers agree that unshared data is the most common case, followed by pairwise shared, followed by multiply shared. Thus the pairwise sharing case may be an important one to optimize. SCI defines optional pairwise sharing optimizations that allow two processors to pass data back and forth without interacting with memory, thus distributing system traffic and reducing activity at the memory controllers.

We made significant changes in the initialization model and in the executable C code that embodies the detailed specifications. In particular, we changed compile-time options to runtime options so that the same code could be used for testing the interaction of nodes implementing different option sets. Also, we significantly improved bit-serial link specifications using material supplied by the Serial HIPPI working group.

The revision process converted all but one of the seven negative votes to affirmative by responding to the concerns expressed. As mentioned earlier, the one remaining negative vote can only be changed by converting the C code to Pascal, which would be unacceptable to the working group.

We redistributed the resulting Draft 2.0 to the balloting body in December 1991 and expect that the standard will receive final IEEE approval early in 1992. Supporting chips should be available within a few months. Dolphin SCI Technology of Oslo, Norway, will offer single-chip SCI interfaces that incorporate transceivers, FIFOs, and cache coherence support. Several versions are planned, for use as a processor, a memory, or an I/O interface. Hewlett Packard is preparing parallel/serial converter chips to interface the Dolphin chips (parallel) to the 1,000-Mbps serial encoding used on optical fiber or coaxial cable. These chips should be available early in 1992.

Related standards projects

The following projects are either used by SCI or form a part of ongoing work related to future SCI developments.

IEEE Std 1212. The Control and Status Register Architecture standard defines the I/O architecture for SCI, Futurebus+ (IEEE Std 896.1 and 896.2-1991), and Serial Bus (P1394). David V. James, Apple Computer, 20525 Mariani Ave., Cupertino, CA 95014, phone 408-974-1321, fax 408-974-0781, dvj@apple.com, chaired the group. Voters approved the draft standard, which was then modified in response to ballot comments. The balloting body received a second and final recirculation, and the IEEE Standards Board granted final approval in December 1991.

IEEE Std 1301. The *Metric Equipment Practice for Micro-*

**We expect SCI, Draft 2.0, to
receive final IEEE approval early
in 1992 and supporting chips to
be available within a few
months.**

computers—Coordination Document is an approved standard (June 1991). This specification defines the generic metric modular packaging family used by the SCI module. Hans Karlsson, Ericsson Telecom AB, TN/ETX/T/F, Stockholm, S-126 25 Sweden, phone +46-8-719-6037, fax +46-8 719 8282, chaired the group.

IEEE Std 1301.1. *The Detailed Standard for a Metric Equipment Practice for Microcomputers Using 2-mm Connectors and Convection Cooling* is also approved (June 1991). This specification details the specific subset of IEEE Std 1301 that is used by the SCI module. Hans Karlsson served as chair. EIA IS-64, February 1991, presently defines the 2-mm connector, but it will become an IEC standard soon. (This connector family is sometimes called Metral, which is DuPont's trademark.)

P1394. The Serial Bus working group is developing a high-speed (10-20 Mbytes/s) serial bus that can be used for low-cost diagnostics (supporting IEEE Std 1149.1 Boundary scan Architecture in large systems) and I/O. The group aims at a very low cost (\$15 per connection, including cable, connector, and interface) bus for use in consumer products. SCI includes a Serial Bus connection in the module power connector. Michael Teener, Apple Computer, 3535 Monroe St., Santa Clara, CA 95051; phone 408-974-3521, fax 408-985-9893, teener@apple.com, chairs the group. Work on this standard should complete in 1992.

P1596.1. The SCI/VME Bridge project is defining a bridge architecture for interfacing VMEbuses to an SCI node. This project provides I/O support for early SCI systems via VME. Products are likely to be available in 1992. Bjorn Solberg, CERN, CH-1211 Geneva 23, Switzerland, phone +41-22-767-2677, fax +41-22-782-1820, bsolberg@dsy-srv3.cern.ch, chairs the group.

The project's main decisions involve the mechanism for mapping addresses between VME and SCI, which versions of VME to support, and how to handle interrupts, mutual exclusion, and cache coherence.

P1596.2. The Cache Optimizations for Large Numbers of SCI Processors project is developing request combining, tree-structured coherence directories, and fast data distribution mechanisms needed for systems with thousands of proces-

sors, compatible with the base SCI coherence mechanism. Ross Johnson, Computer Science Dept., 1210 Dayton Street, University of Wisconsin, Madison, WI 53706, phone 608-262-6617, fax 608-262-9777, ross@cs.wisc.edu, chairs this group.

This working group is developing and extending ideas that came up during the development of the base SCI standard, but which it felt could be postponed to avoid delaying SCI's introduction. That is, the protocols defined by P1596 seem adequate for systems with perhaps hundreds of processors (enough for a year or so) and include hooks for adding these optimizations later.

When a large number of requests is addressed to the same node, the interconnect becomes congested and performance suffers. Certain kinds of requests, such as reads and fetch-and-adds, may be combined in the network to reduce this congestion. Request combining allows several requests to be combined into one when they meet (waiting in queues in the interconnect). All but one of these requests generate an immediate response, which tells the requester to get the data from that one's cache instead.

SCI nodes handle this kind of no-data response already, because it is used in the basic coherence protocol. The remaining request goes forward, eventually resulting in a response that provides the data to that cache, where the other nodes read them. This design spreads out the traffic, reducing congestion.

Note that these immediate responses relieve the interconnect from retaining any information about the combining, which enormously simplifies the process compared to previous implementations.

Once the data become available, the time needed to distribute them to all the requesters becomes important. The linear linked lists of the base SCI standard result in times proportional to the number of requesters, which can be a performance problem in large systems.

Instead of linear lists, the group would like to maintain a tree structure, which could distribute the data in time proportional to the logarithm of the number of requesters.

At first, group members thought it would be impractical to maintain binary trees in the distributed coherence directory because the overhead would be too high. The schemes they had seen others use were not acceptable for SCI. These involved setting lock variables to get mutual exclusion while tree maintenance was done, thus scaling poorly and violating an SCI design principle. (Lock variables also introduce a variety of complications, such as what to do when the process that holds the lock fails.)

Thus the group first considered using approximate or temporary pointers, which would form short cuts along the linear directory lists but gradually become inaccurate as processors rolled out cache lines and so on. Whenever a temporary pointer was used, it would be checked for validity, and the algorithm would drop back to following the (al-

ways valid) linear list when necessary.

But at the August 1991 meeting, Ross Johnson presented a method for maintaining correct trees at all times, without using lock variables and without adding much overhead. Though details need to be worked out and some corner cases need more study, the group feels the remaining questions can be resolved.

Open issues concern the worst case scenarios (when things happen in the worst possible sequence) and how to reduce the likelihood of having these occur.

P1596.3. The Low-Voltage Differential Signals for SCI project specifies low-voltage differential signals suitable for high-speed communication between CMOS, GaAs, and BiCMOS logic arrays used to implement SCI. The object is to enable low-cost CMOS chips to be used for SCI implementations in workstations and personal computers, at speeds of at least 200 Mbytes/s. Gary Murdock, National Semiconductor, 642 Pineview Drive, San Jose, CA 95117, phone 408-721-7269, fax 408-721-7218, chairs this group.

Five current projects support future SCI applications.

Faster signaling requires smaller signals, if edge rates and currents are to be kept reasonable. Smaller signals require differential signaling (or at least their own reference independent of system ground). At first glance, differential signaling seems to cost a factor of two in signal traces and pins. But the real cost is much smaller because far fewer ground pins are needed, far less system noise is created (or picked up), and the higher signaling speeds reduce the number of parallel signals needed.

SPICE modeling shows that we can signal at SCI speeds (2 ns/bit/signal pair) with contemporary CMOS technology. MOSIS (a fast-turnaround prototype chip fabrication service) test chips have already reached nearly this performance. In fact, the hardest part of the problem is how to provide or accept the data at the signaling rate!

The group bases present modeling on a 250-mV voltage swing, centered on 1V. This approach provides a little headroom for common-mode rejection at the receiver, while allowing use of 5V, 3.3V, and eventually 2V technologies.

The working group is choosing certain signal levels and rates to be supported as signal interchange (link) standards for CMOS implementations of SCI. It will also define an 8-bit and possibly a 4-bit link to complement the Std 1596-defined 16-bit and 1-bit links. This work should complete in 1992.

P1596.4. The High-Bandwidth Memory Chip Interface

***As technology advances,
SCI will define new
physical link standards
for higher performance
or lower cost.***

project defines an interface that will permit access to the large internal bandwidth available inside dynamic memory chips. The goal is to increase the performance and reduce the complexity of memory systems by using SCI signaling technology and a subset of the SCI protocols. This work was started by Hans Wiggers of Hewlett Packard Laboratories, and I now chair it. (My address appears at the end of this article.)

A serious problem with present memory systems is the need to use a large number of memory chips in parallel banks to get the bandwidth needed for today's powerful microprocessors. As the capacity per chip increases, the smallest memory configuration with adequate bandwidth reaches a point where it has an unreasonably large capacity (and needlessly high cost). Furthermore, the increments for expansion are too large. We hope to get much higher bandwidth from far fewer chips by using SCI signaling technology. This approach will lower the entry cost for low-end systems and raise the performance of high-end systems.

Designers are considering several models.⁵ One of the most promising uses several RAM chip ringlets attached to a single controller by 8-bit-wide point-to-point links. The name RAM Link is becoming popular for this approach. Details of the signaling are still being worked out, but there seems to be general agreement to use small signal voltages.

Other issues include whether to perform ECC (error checking and correction) in each RAM chip or in the controller. Including it in the RAM chip leaves the details up to the vendor, with possible future technology improvements being incorporated transparently. Including it in the controller is probably the lowest initial cost and gives the system designer the most control. But ECC involves a performance penalty for transmitting the extra information on the links. We could increase link performance by using a 9-bit-wide link, but that would cost pins and power.

Another open question concerns the link protocol to be used. If an SCI bypass FIFO can be incorporated on the RAM chips, we can use a simplified version of the present SCI protocols. If not, we must define a scheduling mechanism that prevents two packets from being transmitted at

once. Predictability issues, such as the effects of refresh or ECC soft-error correction on the RAM chip, complicate scheduling. However, in the November 1991 meeting the group proposed a "designated token" scheduling mechanism that looked very promising. Draft 0.11 was presented at the January meeting.

P1596.5. The Shared-Data Formats Optimized for SCI project specifies data formats for efficiently exchanging data between byte-addressable processors on SCI. SCI supports efficient data transfers between heterogeneous workstations within a distributed computing environment. Current systems require conversions among large numbers of vendor- or language-dependent data formats; specifying a single transfer format greatly reduces the complexity of this conversion problem. In addition to simplifying the data-interchange problem, standard data formats provide a framework for the design of future processor instruction sets and language data types. David V. James, Apple Computer, chairs this group.

The specification defines integer and floating-point sizes, formats, and address-alignment constraints. It supports bit fields as subcomponents of a larger byte-addressable integer datum. Work on this project has just begun, but it should not take long to complete, since much of the groundwork has been done earlier in conjunction with development of the CSR Architecture.⁶ Draft 0.50 was presented at the January meeting.

Future plans

As technology advances, SCI will need to define new link standards. For example, present fiber optic technology is currently expensive at 1,000 Mbps and prohibitive at higher rates. Yet this situation changes rapidly, and SCI applications in high-definition television would be greatly helped by a bit rate at least double this. We will monitor progress in this area.

Similarly, SCI offers enormous application possibilities for slower links. An 8-bit-wide link operating at 250 Mbytes/s or 500 Mbytes/s might be about right for the next-generation personal computers. Perhaps it will be appropriate soon to define a personal computer form factor and signal standard for SCI, based on new CMOS chips or processors with integrated SCI interfaces.

While designing SCI, we learned a lot about how the processor should interact with the interconnect. We are considering how best to spread this information to processor designers, to make multiprocessor systems more efficient. Possibly, this could be a recommended practice, or even a standard, clean, 64-bit RISC architecture optimized for use with SCI.


We have in mind two projects for bridges to Futurebus+ that are currently waiting for the right time to start. One is a very simple bridge to Profile B, an I/O bus with no cache coherence. The other is a general symmetric bridge that includes cache coherence.

THE BASE SCI STANDARD COVERING THE physical signaling, logical protocols, and cache coherence mechanism should be approved by the IEEE Standards Board in March 1992. The first commercial implementer (Dolphin SCI Technology, Oslo, Norway) expects to have working prototypes within a few months of the standard's approval and has promised to make the interface chips available to others.

SCI's performance seems such a large step ahead of the current state of the art in computer buses that it has some difficulty in appearing credible. The best answer to these doubts will be the existence of working silicon, available at reasonable prices, being used in working systems.

The complexity of SCI is approximately the same as that of a split-cycle bus system (like the VAX BI or Futurebus+ or Fastbus with Buffered Interconnects) in small applications. It is much less than that of a bus system for large applications. Nevertheless, relatively few designers have experience with split-cycle bus design issues, and therefore we foresee some need for training as they move to SCI.

SCI has no evident competition in terms of open systems that could hope to deal with the massive computation needs for the next generation of data acquisition, analysis, and general computation. Nor is there any competitor that spans SCI's whole application range: campuswide optical LAN, desktop workstation bus, network shared server, I/O interface, data acquisition system, highly parallel multiprocessor, supercomputer.

For details, or to participate in this work, please contact me. 

Acknowledgments

The US Department of Energy contract DE-AC03-76SF00515 supported this work.

References

1. "SCI-Scalable Coherent Interface, P1596/D2.00 18Nov91," Draft for Recirculation to the Balloting Body, prepared by the P1596 Ballot Review Committee of the IEEE Microprocessor Standards Committee, 1991; copies available from D.B. Gustavson, dbg@slacvm.slac.stanford.edu.
2. S.L. Scott and J.R. Goodman, "Performance of Pipelined K-ary N-cube Networks," Computer Sciences Tech. Report 1010, Univ. of Wisconsin, Madison, Wisc., Feb. 1991.
3. J.W. Bothner and T.I. Hulaas, "Various Interconnects for SCI-Based Systems," *Proc. Open Bus Systems*, VFEA Int'l Trade Association, 10229 No. Scottsdale Road, Suite B, Scottsdale, AZ 85253, 1991, pp.197-202.
4. J.M. Mellor-Crummey, "Concurrent Queues: Practical Fetch-and-Phi Algorithms," Tech. Report 229, Nov. 1987, Univ. of Rochester, Computer Science Dept., Rochester, New York.
5. S. Gjessing, G. Stone, and H. Wiggers, "RamLink: A High Bandwidth Point-to-Point Memory Architecture," *Proc. Compcon Spring 92*, to be published by IEEE Computer Society Press, Los Alamitos, Calif., in 1992.
6. *IEEE Std 1212-1991, Standard for Control and Status Register (CSR) Architecture for Microcomputer Buses*; prepublication copies available from IEEE Service Center, Piscataway, N.J.

Bibliography

- K. Alnes et al., "Scalable Coherent Interface," *Proc. Comp Euro 90*, pp. 446-453.
- K. Alnes et al., "Chip Sets for Scalable Coherent Interface," *Proc. Open Bus Systems*, VFEA Int'l Trade Association, pp. 209-213.
- G. Delp et al., "Memory as a Network Abstraction," *IEEE Network*, July 1991, pp. 34-41.
- S. Gjessing, "SCI Cache Coherence," *Proc. Open Bus Systems*, VFEA Int'l Trade Association, pp. 189-196.
- S. Gjessing and E. Munthe-Kaas, "Formal Specification of Cache Coherence in a Shared Memory Multiprocessor," Informatics Research Report, Univ. of Oslo, Nov. 1991.
- S. Gjessing, S. Krogdahl, and E. Munthe-Kaas, "Formal Specification and Verification of SCI Cache Coherence," *Proc. NIK*, 1989; also available as Informatics Research Report 142, Univ. of Oslo, Norway, 1990.
- S. Gjessing, S. Krogdahl, and E. Munthe-Kaas, "Approaching Verification of the SCI Cache Coherence Protocol," Informatics Research Report No. 145, Univ. of Oslo, Aug. 1990.
- D.B. Gustavson, "IEEE P1596, A Scalable Coherent Interface for Gigabyte/s Multiprocessor Applications," Special Issue, *Trans. on Nuclear Science*, Vol. 36, No. 1, 1989, pp. 811-812.
- D.B. Gustavson, "Scalable Coherent Interface," *Proc. Compcon Spring 89*, IEEE CS Press, pp. 536-538.
- D.V. James, "Scalable I/O Architecture for Buses," *Proc. Compcon Spring 89*, IEEE CS Press, pp. 539-544.
- D. V. James, "SCI Cache Coherence," *Cache and Interconnect Architectures in Multiprocessors*, M. Dubois and S.S. Thakkar eds., Kluwer Academic Publishers, Boston, pp. 189-208.
- D.V. James et al., "New Directions in Scalable Shared-Memory Multiprocessor Architectures: Scalable Coherent Interface," *Computer*, Vol. 23, No. 6, June 1990, pp. 74-77.
- E.H. Kristiansen, "Scalable Coherent Interface," *New Backplane Bus Architectures*, CERN report CN/90/4, Mar. 22-23, 1990, pp. 67-75.
- E.H. Kristiansen, "SCI-to-VMEbus Bridge, IEEE Std. Project," *Proc. Open Bus Systems*, VFEA Int'l Trade Association, pp. 203-208.
- J.M. Mellor-Crummey and M.L. Scott, "Synchronization Without Contention," *Proc. Fourth Int'l Conf. Architectural support for Programming Languages and Operation Systems*, Assoc. of

Call for Papers

**IEEE Micro seeks manuscripts
for its October and December
1992 special issues.**

- **For the October issue on processing hardware for video communication, the guest editor requests submittals on the following topics: ICs for HDTV compression, ICs for HDTV communications, parallel processing systems with real-time video compression capabilities, and multimedia systems with real-time video compression capabilities.**

Submit six copies of manuscripts by April 1, 1992, to E.D. Petajan, AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974-2070; phone (908) 582-3160.

- **The guest editor of the December 1992 special issue requests submittals on special signal processors.**

Submit six copies of manuscripts by April 1, 1992, to John L. Schmalzel, Division of Engineering, University of Texas at San Antonio, San Antonio, TX 78285; phone (512) 691-5515; jls@sun01.sa.utexas.edu.

Computing Machinery, N.Y., pp. 269-278.

J.M. Mellor-Crummey and M.L. Scott, "Algorithms for Scalable Synchronization on Shared Memory Microprocessors," *ACM Trans. on Computing Systems*, Vol. 9, No. 1, Feb. 1991, pp. 21-65.

S.L. Scott, "A Cache Coherence Mechanism for Scalable, Shared-Memory Multiprocessors," Computer Sciences Tech. Report 1002, Univ. of Wisconsin, Madison, Feb. 1991.

J.E. Smith and J.R. Goodman, "Restricted Fetch Operations for Parallel Processing by Gurindar S. Sohi," Computer Sciences Tech. Report 922, Univ. of Wisconsin, Madison, Mar. 1990.

P.J. Woest and J.R. Goodman, "An Analysis of Synchronization Mechanisms in Shared-Memory Multiprocessors," Computer Sciences Tech. Report 1005, Univ. of Wisconsin, Madison, Feb. 1991.



David B. Gustavson is a member of the Computation Research Group at the Stanford Linear Accelerator Center in Palo Alto, California. Previously, he was a member of an elementary particle physics research group, also at SLAC, with primary responsibility for real-time data acquisition systems, data processing, and computer interfacing. He chairs the P1596 Scalable Coherent Interface and P1596.4 RAM Link projects. He also chairs the Fastbus Software working group (IEEE Std 1177) and serves as a member of the Fastbus (IEEE Std 960) hardware design team and its executive committee. He has participated in the development of a variety of other standards, including Control and Status Register Architecture (IEEE Std 1212), S-100 bus (IEEE Std 696), Futurebus (IEEE Std 896), and Floating-Point Arithmetic (IEEE Std 754).

Gustavson received a BS degree in physics and mathematics from the University of Nebraska, studied at the Georg August Universitat in Göttingen, Germany, as a Fulbright Fellow, and received a PhD in high-energy elementary particle physics from Stanford University. He is a member of the IEEE, the IEEE Computer Society, and the Association of Computing Machinery.

Direct questions concerning this article to the author at Stanford Linear Accelerator Center, MS 88, PO Box 4349, Stanford, CA 94309; dbg@slacvm.slac.stanford.edu.

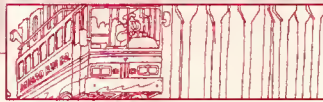
Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

High 150

Medium 151

Low 152



Unix and the Am29000 Microprocessor

Though targeted for use in medium- to high-performance embedded applications, the Am29000 includes several design provisions allowing its use in Unix workstation applications. For example, a scalable interrupt-handling mechanism makes the processor particularly suited to real-time Unix operations. The relevant features discussed here will help users interested in building a Unix system.

Daniel Mann

Advanced Micro Devices

Use of RISC processors in medium- and high-performance embedded applications is growing rapidly. As prices fall, it seems likely that RISC machines will dominate CISC processors as the system of choice for an even wider range of new embedded system designs. A number of companies are already manufacturing processors using RISC principles that offer a better price-performance ratio than the top-end CISC devices.

Many designers have chosen to use Advanced Micro Device's Am29000 processor in complicated embedded applications. The increased use of higher level languages and real-time operating system support services with embedded RISC applications requires engineers to know more about processor features previously more widely used in workstation applications, such as Unix.

Unix was developed on small computers and it makes modest demands on a host processor. However, for good performance Unix requires the processor to provide certain basic facilities. The Am29000 has several features that make it a particularly suitable Unix host. For example, the architecture is scalable yet has features for maintaining code compatibility.

Our processor's interrupt-handling mechanism services devices without incurring a built-in exception-processing sequence. This is of particular interest to implementers of Unix systems that are also constrained with real-time events. Users are free to design the necessary interrupt-handling processor environment for maximum efficiency.

Among the Am29000's features of particular

interest to designers of high-performance Unix systems are

- data and instruction caching,
- memory management,
- multiple data transfer instructions,
- freeze-mode operation,
- multiprocessor support,
- floating-point support, and
- flexible memory systems.

A detailed discussion of the Am29000's architecture^{1,2} is beyond the scope of this article, as is a discussion of Unix internals.^{3,4} Rather, I point out features that make our processor a good Unix host.

C calling sequence

Making a subroutine call on a processor with general-purpose registers is expensive in terms of time and resources. Because functions must compete for register use, registers must be saved and restored through register-to-memory and memory-to-register operations. For example, a C function call on Motorola's MC68000 processor (see Table 1 on page 24) might use the statements

```
char bits8;  
short bits16;
```

```
printf("char= %c short=%d", bits8, bits16);
```

After they are compiled, they generate the following assembly-level code:

Table 1. MC68000 instructions.

Instruction	Comment
MOVE.W saddr,daddr MOVE.W -4 [A6], D0	Move 16 bits of data from saddr to daddr. Source address is register indirect with displacement. Destination address is data register direct. The word at memory location -4 relative to the current frame pointer (A6) is copied into data register D0.
MOVE.B saddr, daddr	Move 8 bits of data from saddr to daddr.
MOVE.L saddr, daddr	Move 32 bits of data from saddr to daddr.
EXT.L data_register	Extend the sign of 16-bit data to 32-bit register size.
PEA L15	Push address L15 onto the stack (A7).
JSR _printf	A jump to subroutine _printf is taken. The current PC value is first pushed onto the stack (A7).
LEA 8 [A6], A0	The address of the data object located 8 bytes above the current frame pointer (A6) is loaded into address register A0.
LINK A6,#-32	Push the frame pointer (A6) onto the stack. Then copy the stack pointer (A7) to the frame pointer (A6). The stack pointer (A7) is then lowered by 32 bytes. This instruction takes several cycles and is used in a procedure prologue.
UNLK A6	The frame pointer (A6) is copied to the stack pointer. A new frame pointer is then popped out of the stack. This instruction takes several cycles and is used in a procedure epilogue.
RTS	The address value for a subroutine return is popped out of the stack (A7) and loaded into the PC.

L15: .ascii "char= %c short=%d"

```

MOVE.W  -4 [A6], D0    ;stack bits16 variable
EXT.L    D0
MOVE.L   D0, -[A7]
MOVE.B   -1 [A6], D0    ;stack bits8 variable
EXTB.L   D0
MOVE.L   D0, -[A7]
PEA      L15            ;stack text string ptr
JSR      _printf
LEA      12 [A7], A7    ;repair stack pointer

```

This assembly listing shows how parameters pass via the stack to the function being called.

The LINK instruction copies the stack pointer A7 to the local frame pointer A6 upon entry to a routine. The parameters passed and local variables in memory are referenced relative to register A6.

To reduce future access delays, the system normally copies data to general-purpose registers before using it. For instance, using a memory-to-memory operation when moving data from the local frame of the function call stack would reduce the number of instructions executed. However, these are CISC instructions that require several machine cycles before completion.

In the example, the C function call passes two variables, bits8 and bits16, to the library function printf(). The following assembly code shows part of the printf() function for the MC68000:

```

_printf:
    LINK A6, #-32 ;local variable
                space
    LEA 8 [A6], A0 ;unstack string
                pointer
    ...
    UNLK A6
    RTS

```

Several multicycle instructions are required to pass the parameters and establish the function context. Unlike the variable instruction format in the MC68000, the Am29000 has a fixed 32-bit instruction format (see Figure 1). The same C statements compiled for the Am29000 (see Table 2) generate the following assembly code for passing the parameters and establishing the function context:

```

L1: .ascii "bits8=%c bits16=%d"
    const lr2,L1
    consth lr2,L1
    add lr3,lr6,0 ;move bits8 and bits16
    add lr4,lr8,0 ;to bottom of the
                ;activation record
    call lr0,_printf ;return addr in lr0

```

Op code	Operand C	Operand A	Operand B
8 bits	8 bits	8 bits	8 bits

Figure 1. Am29000 fixed 32-bit instruction format.

Register stack. We define a register stack in an assigned area of memory to pass the parameters and allocate working registers to each procedure. The register cache replaces the top part of the register stack, as shown in Figure 2 on page 26.

The global registers *rab* and *r1b* point to the top and bottom of the register cache. Global register *rsp* (also known as *gr1*) points to the top of the register stack. The register cache, or stack window, moves up and down the register stack as the stack grows and shrinks. Use of the register cache allows data to be accessed through local registers at high speed. On-chip triple-porting (two read ports and one write port) enables the register stack to perform better than a data memory cache, which cannot read and write in the same cycle.

Activation record. Our processor does not apply push or pop instructions to external memory. Instead, each function is allocated an activation record in the register cache at compile time. Activation records hold local variables and parameters passed to the function.

The caller stores its outgoing arguments at the bottom of the activation record. The called function establishes a new activation record below the caller's record. The top of the new record overlaps the bottom of the old record, so the outgoing parameters of the calling function are visible within the called function's activation record.

Although the activation record can be any size within the limits of the physical cache, the compiler will not allocate more than 16 registers to the parameter-passing part of the activation record. Functions that cannot pass all their outgoing parameters in registers must use a memory stack for additional parameters (global register *msp* points to the top of the memory stack). This happens infrequently, but it is required for the parameters that have their address taken. Data parameters at known addresses cannot be supported in register address space because data addresses always refer to memory and not to registers.

The following code shows part of the `printf()` function for the Am29000 processor:

```
_printf:
    sub    gr1, gr1, 16    ;function prologue
```

Table 2. Am29000 instructions.

Instruction	Comment
const reg, value	The value is placed in the selected register. This is a data-direct instruction. Operand fields A and B hold the 16-bit constant.
const lr2, L1	The lower 16 bits of address L1 are placed in local register lr2. The high 16 bits of lr2 are cleared.
consth lr2, L1	The high 16 bits of address L1 are placed in the high 16 bits of local register lr2.
add des, srcA, srcB	The two source operands A and B are added, and the result placed in the destination register.
add lr3, lr6, 0	Zero is added to the lr6 value, and the result is placed in local register lr3. This is effectively a register-to-register move operation.
call lr0, _printf	A jump to subroutine <code>_printf</code> occurs in the cycle following the current cycle. The current PC address plus 8 is placed in local register lr0. This is effectively a subroutine call with the destination address obtained by adding the current PC value and the 16-bit value formed by operand fields A and B. The Am29000 implements delay-slot branching, thus it always executes the instruction following a branch instruction. Direct 32-bit address procedure calls are supported with the CALLI instruction.
jmpir lr0	The value in local register lr0 is loaded in the PC. Execution at the destination of a jump commences after the instruction following the JMPI has executed. This instruction implements a subroutine return.
asgeu V_SPILL, gr1, rab	This is a conditional trap instruction. Such instructions are used in procedure prologue and epilogues to check whether cache spilling or filling is required. Trap number V_SPILL is taken if the assertion that the global register gr1 is greater than or equal to global register rab is false.

```
asgeu    V_SPILL, gr1, rab    ;compare with
                                ;top of window
add      lr1, gr1, 36        ;rab is gr126
...
jmpir    lr0                 ;return
asleu    V_FILL, lr1, r1b    ;compare with
                                ;bottom of
                                ;window gr127
```

The register stack pointer *rsp* points to the bottom of the current function's activation record. All local registers are referenced relative to *rsp*. Four new registers are required to support the function call shown, so *rsp* is decremented 16 bytes.

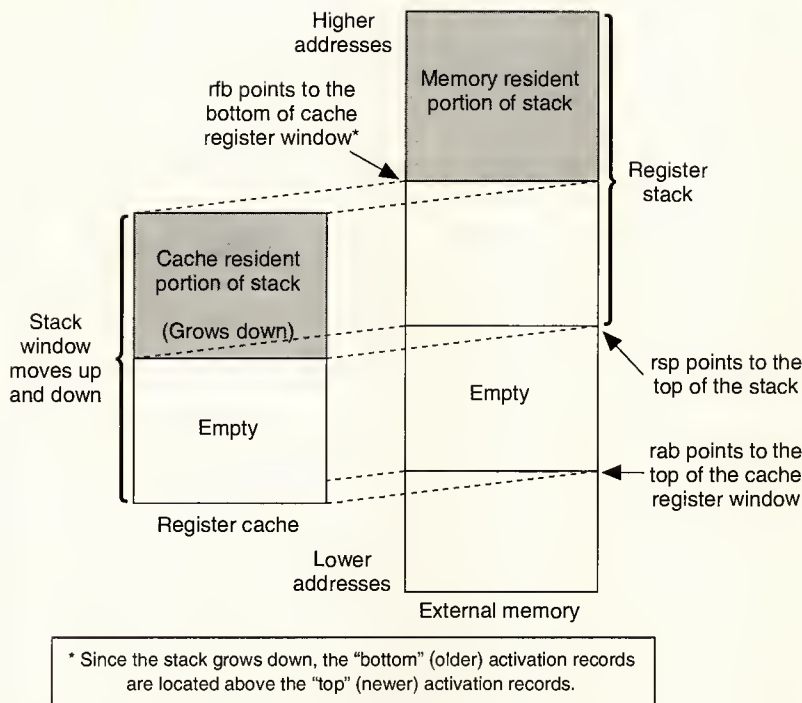


Figure 2. Register stack window.

Rsp performs a role similar to the MC68000's A7 and A6 registers except that it points to data in high-speed registers, not data in external memory.

The compiler reserves local registers lr1 and lr0 for special duties within each activation record. lr0 contains the execution starting address when it returns to the caller's activation record. lr1 points to the top of the caller's activation record. The new frame allocates local registers lr2 and lr3 to hold printf function local variables.

As Figure 3 shows, the positions of five registers overlap. The three printf() parameters enter from lr2, lr3, and lr4 of the caller's activation record and appear as lr6, lr7, and lr8 of the printf() function activation record.

Spill and fill. If not enough registers are available in the cache when it moves down the register stack, then a V_SPILL trap is taken, and the registers spill out of cache into memory. Only procedure calls that require more registers than currently are available in the cache suffer this overhead.

Once a spill occurs, a fill (V_FILL trap) can be expected at a later time. The fill doesn't happen when the function call causing the spill returns, but rather when some earlier function that requires data held in a previous activation record (just below the cache window) returns. Just before a function returns, the lr1 register, which points to the top of the caller's activation record, is compared with the pointer to the bottom

of the cache window (rfb). If the activation record is not stored completely in the cache, then the fill overhead occurs.

Performance. The register stack improves the performance of call operations because most calls proceed without memory access. The register cache contains 128 registers, so very few function calls or returns require register spilling or filling.

Because most of the data required by a function reside in local registers, there is no need for elaborate memory-addressing modes, which increase access latency. The function call overhead in the Am29000 consists of a small number of single-cycle instructions; the MC68000 requires a greater number of multicycle instructions.

Context switching

Context switching occurs when one process gives up control of the CPU without terminating, and another process, which had previously given up control, resumes executing. When this happens, the state of the processor being used by the process (the context) must be saved, and the context of the other process must

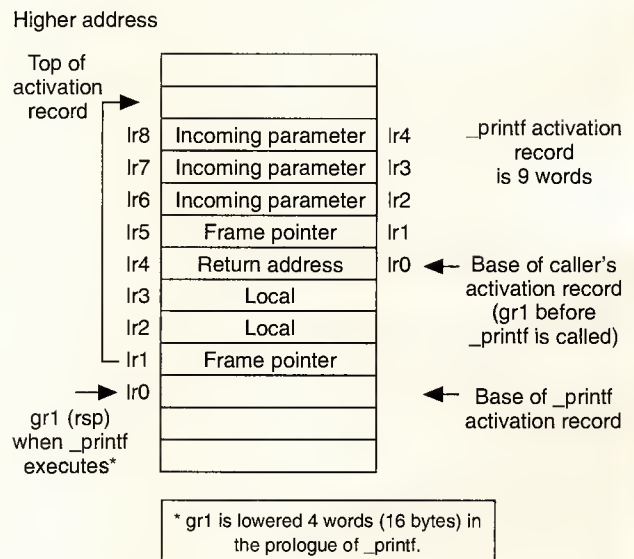


Figure 3. Overlapping registers.

be restored. Other than allowing multiple processes to share the same CPU, this saving and restoring of context perform no useful work. If context switching is performed frequently, it should be a low-overhead operation.

One result of having a large register file is an increased context-switching time. Approximately half of the 128 local registers and about 32 of the global registers contain data for the user's process. About 96 memory writes and 44 memory reads are required to store and reload these registers with a new context. However, with single-cycle, burst-mode access, the register save and restore time can be as short as 5.6 μ s. Note that the context reloading for synchronously saved process states is faster than context saving because only the activation record of the currently active procedure need be restored. The number of local registers requiring restoring decreases from approximately 64 to, typically, 12.

Memory access time dominates the total context switch time. In a low-cost system with no external data cache, this is most apparent. When using five-cycle read/write memory, the memory access time is 28 μ s. In practice, any increase in context-switching time due to the large register set is more than offset by the savings in normal execution. This is because context switches (30 to 60/s) are less frequent than system calls (150 to 250/s) or subroutine calls (about 350,000/s). The Am29000 can switch context in as little as 10 μ s, about 45 times faster than the VAX 11/780.

System calls

System calls are trusted library functions that execute with kernel mode permissions to access resources otherwise unavailable to the user. To users, a system call looks like a normal C function call. But when a user program executes a system call, it leaves user mode and enters kernel mode.

Permission levels. When a system call is made, the selected function executes a trap instruction to change the processor status. After the trap is taken, the process operates with kernel mode permissions. It is usually necessary for the operating system to copy the calling parameters from the user mode register stack to the kernel data space, because system call functions receive their parameters from kernel data memory space. To enable system functions to appear as ordinary C functions, the register stack parameters are transferred into a kernel data structure known as the *upage* by a system call dispatcher routine.

The kernel mode permissions of system calls may not access data that the user would not normally be able to access. When moving data between user space and kernel space, many processors use assembly level instructions to manipulate the memory management unit (MMU) or other protection hardware. This allows the kernel to access the user data space with the user's permissions rather than kernel's permissions.

The usual set of Unix functions available includes a fetch user byte, `fubyte(a)`, and a store user byte, `subyte(a,v)`. As an

example, the statement

```
result = fubyte(addr);
```

fetches a byte of data from user address `addr` and stores it in the variable `result`. This variable is in kernel data space, and the access of the user data space occurs with the user's permissions. The `subyte(a,v)` function moves data in the opposite direction, from kernel space to user space. This routine is required to modify data structures in the user's address space.

Protection boundaries. Our processor efficiently copies data across protection boundaries in a Unix implementation. System call parameters are not located in the user's data memory space but in the register cache, where they don't result in a heavy overhead.

After the system call trap is taken, call parameters can be copied from the registers into the *upage* without any permission problems because the register cache is wholly owned by the user process. There is no risk of the kernel accessing data belonging to another user. The store multiple instruction moves data quickly from registers to memory.

The processor handles system call return values the same way as any other function call return values. Global registers from `gr96` through `gr111` hold the first 16 words of a function return value. Any other data in the user's data space that the system call uses or updates can be accessed by setting the user access (UA) bit in load and store (LOAD, STORE) instructions. The UA bit allows programs executing in kernel mode to emulate user mode access. Doing so enables functions such as `subyte()` to be implemented inexpensively, particularly since the UA bit can be used with the load and store multiple (LOADM, STOREM) instructions.

Overhead. Because few accesses to data memory are required, the overhead of switching from user mode to kernel mode is small. In kernel mode, separate memory and register stacks are used, but they are still accessed via global registers `rab`, `r1b`, `mzp`, and `rsp`. Upon entering kernel mode, these registers contain user mode values which are then stored in the *upage* (part of the context save) and replaced with addresses of kernel-mode memory and register stacks. The register stack is not pushed to memory, but rather the kernel register stack is added to the end of the cache.

If the barriers between user and kernel modes are high and secure, the computation required for a mode change may far exceed the cost of the actual requested operation. In cases of extremely poor design, the simplest system calls may require hundreds or even thousands of overhead instructions.

In keeping with RISC concepts, we structured the system call operation to ensure the minimum overheads at each stage. As an example of the results achieved, `getpid()` executes in about 10 μ s, depending on the memory architecture. The same function call has been measured at 400 μ s on a VAX 11/780. The system call mechanism results in an overhead that is one

hundredth of that incurred by a VAX 11/780.

Interrupt handling

As a RISC processor, the Am29000 limits the mechanism for handling interrupts and traps so system designers need not be concerned about when, where, or how much state is saved. State saving is not built into the processor interrupt mechanism but is left for the programmer to implement. Interrupts use a vector table to select the required service routine. After nine cycles (0.36 μ s) or less, the processor executes the service code. A freeze mode makes this possible.

Freeze mode. When an interrupt or trap occurs, the processor enters freeze mode. The hardware execution context described by critical CPU registers is not saved, but the state of these registers is frozen and cannot be updated until an interrupt return is taken or freeze mode is turned off. Interrupts that do not require the use of processor special registers can be serviced in freeze mode. This allows a fast interrupt service response. In fact, it is practical to implement a software cache-controlled reload in low-cost systems.

Certain critical registers are required for the execution of C code routines. If the interrupt-handling code is just assembly glue used to reach a C code routine, then the critical registers must be saved explicitly before freeze mode is turned off. Users can tailor the form of the register context stacking procedure to meet special needs. This facility can prove useful in optimizing system operation.

Interrupt servicing. Our processor's high-speed operation enables interrupts to be handled by a Unix implementation without assigning a priority order. When an interrupt is detected, it will be serviced immediately if no other interrupts are currently being serviced. Otherwise, the interrupt is added to the end of a queue. The processor continually tries to empty this queue and return to executing user processes.

When the first interrupt occurs, possibly initiating a queue, the processor may be in user mode or kernel mode. Although each process executes the same kernel code, each process has its own kernel stack for kernel mode function calls and data. In user mode, the kernel memory stack stores data during the interrupt service routines. If the interrupt occurs in kernel mode, then a separate shared interrupt stack is used.

Although I have described only one stack, the processor contains two: one for conventional memory data and one for register data that has spilled out or swapped out of the register cache.

Virtual address space

Unix is a multiprocess system, with several processes existing in different areas of system memory at the same time. Programs may be loaded at some memory location and then change location during execution due to being swapped out and back in again. Programs are usually intended to execute from virtual address zero, but the actual physical memory

used by the program is dependent on the relocation function of the MMU. The program, or parts of the program, may get swapped to new physical locations. But because of the address translation of the MMU, they always appear to be at the same virtual address.

On-chip MMU. We built a standard MMU into the Am29000 chip to lower system cost and maintain compatibility among 29000 users. Most other processors require expensive external hardware to translate addresses. In such cases, designers sometimes prefer to develop their own MMUs. This can lead to incompatibilities with code developed for other designs.

Translations. The Am29000 uses a 64-entry translation look-aside buffer (TLB) to perform address translations. The TLB translates the most frequently used instruction and data memory pages and reflects the information contained in more extensive tables in memory. Data addresses are translated during each execute processor stage. Instruction addresses are translated whenever a jump is taken, achieving virtual memory support without any access delays.

Because of the TLB's limited size, there is a chance that an access will be requested for a memory page not currently translated—a TLB miss. When this occurs, the special register *lru* (least-recently used TLB entry) selects a TLB register for replacement with new translation data.

The TLB registers are not updated automatically by hardware. When a TLB miss occurs, a trap to a software routine that maintains TLB entries is performed. Using this method to update the TLB allows a variety of memory management techniques to be implemented. Systems that rely on hardware to update the TLB registers do not achieve this level of flexibility. The technique is particularly well suited to the emulation of missing hardware accessed in virtual address space. A TLB register can be updated in two Am29000 cycles (one for each word of a TLB entry). This keeps down the software overhead of reloading TLB. Depending on page-table layout, a TLB reload needs 25 to 40 cycles after a page miss trap occurs.

Memory access protection

In any multiprogramming system, especially one in which users are developing, debugging, and testing software, process *a* should be forbidden access to the data of process *b* if accessing that data would impair process *b*. In addition, some operating system data should be protected even from read-only access.

Reducing memory usage and permitting efficient multiprocess cooperation requires controlled sharing of regions of memory and other resources. For many programs, a large portion of executable code consists of the system-provided library subroutines. For small programs, the library code used by the program can be larger than the program itself, particularly if the program makes heavy use of standard I/O routines. Enhanced Unix systems provide shared libraries so

only one copy of the library code is required for all the processes in the system.

Protection violation checking. The TLB hardware checks for protection violation. Each TLB entry contains a 6-bit field that controls access to the associated page, which is assigned to a particular user through a task identifier. Permissions can be set to separately enable reading, writing, and execution of page data. The processor mode and user ID are checked with the access permissions for the virtual memory page accessed. A software trap occurs if access is not permitted. Including user identifiers in each TLB entry enables Unix to switch processes without clearing all TLB entries. Performance is improved because valid TLB entries are likely to remain and be reused when their associated process is swapped back in.

Protection control. The Am29000 offers considerable access control within the chip. Other systems that use external hardware to achieve a similar function incur a greater system cost. Also, by including the necessary hardware in our design, we achieve a standard that cannot be ensured when designers build their own protection systems.

Cache support

The Am29000 incorporates an on-chip branch target cache (BTC) memory. If the target instructions of a branch are found in the cache, the branch executes in one cycle, causing no stalling of the processor pipeline. For sequential instruction access, an instruction prefetch buffer ensures that instructions are fetched up to four cycles ahead of their execution, enabling single-cycle instruction execution at high speeds.

The BTC memory contains the target instruction sequence for 32 branches. Simulation results show this is sufficient for 60 percent of the branch instructions. If a BTC memory miss occurs, the hardware automatically updates the cache with the new target instruction sequence. In such cases, the execution pipeline stalls for one cycle plus the number of cycles required to access the instruction memory. Having an on-chip instruction cache enables users to implement low-cost systems that achieve high speeds.

Data caching. The Am29000 processor caches data via a large register file to reduce the number of memory loads and stores. This reduces the requirement for an external data cache. Of the 192 general-purpose registers, 128 serve as a register cache. (See Figure 4.) This cache stores data variables and function call parameters by using an overlapping stack frame technique.

On occasion the cache becomes full and some data spills out to memory. Simulation results of programs such as *nroff* show this is necessary in only 0.1 percent to 0.5 percent of all calls. Since calls typically constitute 1.2 percent of all instructions used, and spilling out and filling back typically require only 25 to 30 cycles, the overhead is low.

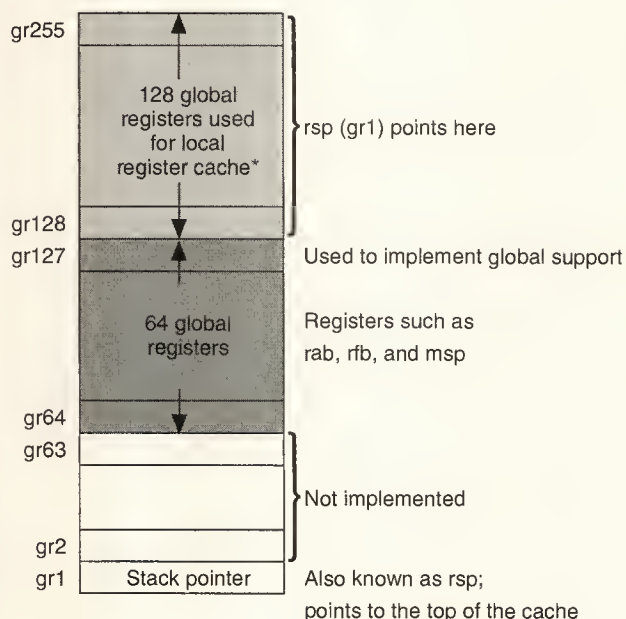
Accessing data in the register file does not suffer the delays

encountered in accessing external data, even if a high-speed cache is used. Thus the Am29000 processor is well suited for even higher speed devices in future technologies. The register file has a further advantage in that it is triple-ported. It accesses two sources in one cycle, while a previously computed result is written back. Subsequent accesses take one cycle, using burst mode.

On-chip instruction caching. BTC memory caches user mode and kernel mode instructions. The cache tag logic works with physical addresses, possibly after they have gone through the virtual-to-physical address translation service of the TLB. The BTC memory entries need not be invalidated on a context switch. Keeping entries valid for future reuse improves performance. However, when the kernel decides to swap user process pages in or out of disk memory, instructions at cached physical memory locations may change, invalidating the cache.

The newer Am29030 processor contains an 8-Kbyte instruction cache. The addition of the cache was made possible by the higher levels of integration not possible when the Am29000 was introduced. Research shows that for cache sizes above 4 Kbytes, a conventional cache provides superior performance to the BTC memory method.⁵

Global registers



* Registers in the cache are normally referred to as local registers and are accessed relative to gr1. Gr1 points to local register lr0. Local register lr1 is located at register address gr1 + 4. All registers are 32 bits wide.

Figure 4. Am29000 register stack.

Because the BTC memory reduces initial instruction access latency rather than improving memory access bandwidth, the Am29000 uses a separate bus to support concurrent instruction fetching. The Am29030 does not require separate buses for instruction and data memory because the on-chip cache offers a statistical improvement to the memory bandwidth. It maintains sufficient instruction memory bandwidth without the support of a dedicated bus.

Multiprocessor Unix

Our processor supports on-chip multiprocessor systems without the need for large amounts of external hardware. It also supports the mechanisms to ensure processor resource sharing and signaling.

An external coprocessor interface extends its execution unit with special-purpose instructions. If the coprocessor hardware is not present, then a coprocessor exception trap is taken. The processor emulates the coprocessor operation for hardware development and code compatibility.

Multiprocessor cache support. Each TLB entry contains 2 bits to control external cache operation. Cached pages can be marked selectively as not-to-be-cached, local, or shared. This enables the operating system to inform the cache of the correct action to take, and it results in reduced data traffic on the shared-memory bus giving access to the cache.

A load and lock (LOADL) instruction for device and memory interlocks activates the LOCK output pin during the address cycle of the access. Multiprocessor hardware uses the lock signal to delay access of other processors requesting the same memory or I/O facility.

The load and set (LOADSET) instruction implements a binary semaphore. The operation of writing a memory semaphore location to True after reading the location into a general-purpose register, cannot be interrupted.

Floating-point support

Unix is a product of the academic and scientific environment, which means that a number of its user application programs require floating-point support. The newer Am29050 processor supports floating-point instructions directly. However, the Am29000 processor instruction set contains a number of floating-point instructions that are not directly supported.

When a floating-point instruction is encountered, it traps to a handler that sends the appropriate instruction and data to an optional coprocessor, the Am29027, and moves the results back to the Am29000. Without the coprocessor, the Am29000 calls software routines to perform the instruction evaluation. This latter method is much slower than using a coprocessor. But by using the trap

method, code developed for the Am29000, which may or may not have an Am29027 coprocessor available, will run directly on future Am29000 processors (and at much greater speed).

A library of software routines has been developed that makes extensive use of the floating-point instructions to perform functions. We reserved 15 user-mode-accessible global registers for math library use to achieve good performance for these functions. We can reduce the number of global registers by substituting memory locations, but doing so reduces performance. Also, all library users must agree on register assignment if the library is to be shared.

System costs

The on-chip MMU functions enable the Am29000 to operate without caches and maintain good speed. Table 3 draws on information in the "Am29000 Performance Analysis" document.⁶ The table compares AMD's 29000 processors running at 25 MHz against Sun processors and the VAX 11/780.

The benchmarks for the Am29000 in Table 4 are for systems with separate instruction and data memory systems. The Am29000 has separate instruction and data memory buses and a shared address bus. The need to implement a memory system for instructions and a separate system for data can be avoided by connecting the instruction and data buses. Doing so makes the Am29000 processor bus system appear more like a conventional processor bus system. However, sharing a single memory system reduces performance by, typically, 15 percent. The use of video memory (video DRAM) also avoids the need for two memory systems, as the video-out port can be connected to the instruction memory bus, and the random access port to the data bus.

Table 4 gives the approximate costs of a CPU and memory devices. It also shows the cost for a memory system based on 4 Mbytes of DRAM with an additional 0.5-Mbyte SRAM used to implement a cache. Studies show that software-controlled caches can efficiently use high-speed memory with a minimum of hardware complexity and cost.⁷ The lightweight interrupt-handling capability achieved by the Am29000 processor operating in freeze mode is well suited to implementing a soft cache control mechanism.

Table 3. CPU speeds.

Benchmark	Am29000		Am29030	Other processors		
	SRAM	DRAM	DRAM	Sun 4	Sun 3	VAX
diff	20.35	12.02	20.86	8.50	3.56	1.0
grep	14.81	9.42	18.54	7.00	3.00	1.0
nroff	14.92	10.78	18.31	7.50	2.30	1.0
Dhrystone 2.0	23.00	15.30	19.69	11.71	2.65	1.0

Table 4. CPU and memory costs.

Feature	Am29000					Am29030
	SRAM	DRAM	Soft cache	Video DRAM	DRAM	DRAM
Instruction memory	4 Mbytes	4 Mbytes	4-Mbyte DRAM, 0.5-Mbyte SRAM	4 Mbytes	4 Mbytes	4 Mbytes
Data memory	4 Mbytes	4 Mbytes	Shared	Shared	Shared	Shared
Cost	\$2,792	\$424	\$352	\$448	\$264	\$262

The Am29030, unlike the Am29000, shares a bus for instruction and data memory and thus does not require separate memory systems. It achieves higher performance via its on-chip, 8-Kbyte instruction memory cache. The Am29050, which is currently the only 29K family member that directly executes floating-point instructions, is pin compatible with the Am29000 processor and thus supports the three-bus architecture.

Because our processor uses only one address bus, the number of pins and the system costs are reduced. The one-address bus technique is successful because of burst-mode addressing and the large register cache. The register file greatly reduces the need to access data in external memory. Burst-mode addressing enables instructions to be fetched in sequence without sending a new address for each access. The address bus is only required to establish the address of the first instruction in a nonsequential instruction fetch. Video DRAM or conventional RAM interleaved with a small amount of support circuitry, such as an address counter, can be used to implement burst-mode memory addressing.

Contention for the address bus for both instruction and data access is rare. By the very nature of the instruction stream, a jump instruction cannot occur on the same cycle as a load or a store operation, thus the address bus is inherently used for instruction and data addressing on different cycles.

THE AM29000 PROCESSOR'S REDUCED INSTRUCTION set results in concise, efficient executable code. Each processor in this family sustains a performance level of 17 VAX MIPS or higher. In addition, the processors have several design features that make them particularly suitable for efficient Unix implementation, both in terms of operating speed and support peripherals required.

Among these is a scalable methodology that frees the system implementer from a fixed-price, fixed-architecture approach. Because the architectural features required to support Unix are packaged along with the processor, the Am29000 processor is particularly suited to use in low-cost systems. ■

References

1. *The Am29000 User's Manual*, Advanced Micro Devices, Sunnyvale, Calif., 1989.
2. M. Johnson, "System Considerations in the Design of the Am29000," *IEEE Micro*, Vol. 7, No. 4, Aug. 1987, p. 28-41.
3. M.J. Bach, *The Design of the Unix Operating System*, Prentice-Hall, Englewood Cliffs, N.J., 1986.
4. S.J. Leffler et al., *The Design and Implementation of the 4.3 BSD Unix Operating System*, Addison-Wesley, Reading, Mass., 1989.
5. M.D. Hill, "Aspects of Cache Memory and Instruction Buffer Performance," PhD Report UCB/CSD 87/381, Electrical Engineering and Computer Science Dept., Univ. of Calif., Berkeley, Calif., Nov. 1987.
6. T. Olsen, "Am29000 Performance Analysis," technical report 10621A, AMD, 1988.
7. D.R. Cheriton et al., "Software-Controlled Caches in the VMP Multiprocessor," report STAN-CS-86-1105, Dept. of Computer Science, Stanford Univ., Stanford, Calif., Mar. 1986.



Daniel Mann is a senior member of the technical staff at Advanced Micro Devices' Am29000 Support Products Division. He works on operating systems and debugger issues. Mann earned BS and PhD degrees in electronic engineering from RGIT in Aberdeen, Scotland. He is a member of the IEEE Computer Society.

Address questions concerning this article to the author at Advanced Micro Devices, 5204 E. Ben White Blvd., MS 573, Austin, TX 78741, or via e-mail at daniel.mann@amd.com.

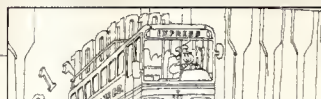
Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 153

Medium 154

High 155



Hardware Requirements for Neural Network Pattern Classifiers

A Case Study and Implementation

A special-purpose chip, optimized for computational needs of neural networks, performs over 2,000 multiplications and additions simultaneously. Its data path is suitable particularly for the convolutional architectures typical in pattern classification networks but can also be configured for fully connected or feedback topologies. A development system permits rapid prototyping of new applications and analysis of the impact of the specialized hardware on system performance. We demonstrate the power and flexibility of the processor with a neural network for handwritten character recognition containing over 133,000 connections.

Bernhard E. Boser

Eduard Sackinger

Jane Bromley

Yann leCun

Lawrence D. Jackel

AT&T Bell Laboratories

Neural networks are rapidly gaining acceptance as powerful and versatile tools for pattern classification.^{1,2} However, the widespread use of neural network classifiers remains contingent on the availability of powerful hardware to provide adequate speed. Such hardware is particularly important as the computational requirements of neural network algorithms are quite different from the high-precision processing for which general-purpose computers are optimized. Typical neural network problems involve huge amounts of low-resolution and possibly redundant data and require a correspondingly high number of low-precision arithmetic operations to be performed.

Special-purpose VLSI (very large-scale integration) processors let us overcome neural network implementation problems. With their regular structure and the small number of well-defined arithmetic operations, these networks are well matched to integrated circuit technology. The high density of modern technologies lets us implement a

large number of identical, concurrently operating processors on one chip, thus exploiting the inherent parallelism of neural networks. The regularity of neural networks and the small number of well-defined arithmetic operations used by neural algorithms greatly simplify the design and layout of VLSI circuits.

But processing speed is not the only constraint on neural network hardware design; neural network classifiers benefit from a highly structured topology with local receptive fields. Of particular importance are convolutional architectures in which neurons with identical weights process different parts of the input or internal state. This topology builds into the network knowledge about locality of data, improving recognition performance. At the same time it lets us multiplex neurons with identical sizes and realize the large networks required for difficult classification tasks within the density limitations of current VLSI technology. The high speed of VLSI technology—five orders of magnitude greater than that of

natural neurons—compensates for the loss of computing speed resulting from partial serial processing in such an implementation.

Matching the arithmetic precision of the hardware to the requirements of neural networks is crucial for an efficient hardware implementation. Neural networks are often quoted for their low-resolution requirement, which lends itself well to analog implementation. Experiments, however, show that the precision requirements of neurons within a single network vary. Specifically, research indicates that higher accuracy is often needed in the output layer, for example, for selective rejection of ambiguous or otherwise unclassifiable patterns. This situation can be handled with a hybrid architecture, which evaluates the bulk of the network with low-resolution analog hardware but implements selected connections on a digital processor with higher accuracy.

Based on these considerations, we designed and fabricated ANNA (Artificial Neural Network ALU), a special-purpose chip for neural network pattern classification.³ The chip features 4,096 individual synapses that can be multiplexed to implement networks with several hundred thousand connections. We can program the number of synapses per neuron to values between 16 and 256; the number of neurons varies accordingly between 256 and 16. Implementable network topologies include fully connected architectures with or without feedback, local receptive fields, and TDNNs (time-delay neural networks) or higher order convolutional connections. Depending on the network topology, the chip can sustain a performance of up to 5×10^9 connections per second (C/s). Arithmetic operations take place with 6-bit resolution on the weights and 3-bit resolution on the states.

Because of the high speed, parallelism, low resolution, and other characteristics of ANNA, which differ considerably from those of general-purpose computers, this hardware must be readily accessible to the network designer, so new applications can be prototyped easily. A development system, consisting of a PC or VME board containing an ANNA chip and a digital signal processor (DSP), addresses these needs. We download the topology and weights of a network into the VME board, and the DSP issues control commands to ANNA. The DSP also preprocesses and trains the networks and processes operations that require higher precision than that of ANNA.

We selected an optical character recognition neural network to test and demonstrate the flexibility and power of the neural network chip.¹ This network identifies handwritten digits from a 20×20 -pixel input image and employs neurons with local receptive fields as well as a fully connected layer. The network with over 133,000 connections fits on one ANNA and is evaluated at a rate in excess of 1,000 characters per second, which constitutes a speedup of two orders of magnitude over a DSP-based implementation. Despite the low resolution of the chip, the error rates of the neural network

***Our OCR neural network with
133,000 connections identifies
handwritten digits at 1,000 cps
and fits on one ANNA chip.***

processor and DSP implementation are very similar at 5.3 percent and 4.9 percent. For comparison, the measured human performance on the same database is 2.5 percent errors.

Neural network hardware

Artificial neural networks that solve difficult problems in areas such as speech recognition and synthesis, or pattern classification, consist of thousands of neurons with tens or hundreds of inputs each. Every neuron computes a weighted sum of its inputs and applies a nonlinear function to its result. Architectural parameters, such as the number of inputs per neuron, and each neuron's connectivity vary considerably within a network, and from application to application. A special-purpose neural network processor must be flexible and powerful enough to accommodate a wide range of applications. At the same time, the requirements must be carefully balanced and the special nature of the task exploited to bring an efficient implementation within reach of today's technology.

We can distinguish two phases of operation in many neural network applications. During the learning phase, the topology and weights of the network are determined from a labeled set of examples using a rule such as backpropagation,⁴ or a network-growing algorithm.⁵ In the subsequent retrieval or classification phase, the network parameters are fixed.

The network recognizes patterns based on information stored in the architecture and weights during training. Since the computational and infrastructure requirements (training database) during the learning phase are considerably more complex than those for classification, efficiency considerations call for separate hardware for learning and retrieval. Network parameters determined during learning are downloaded into processors specialized for the classification task. This approach, which we focus on here, contrasts with implementations of neural network processors with on-chip learning.^{6,7} Those circuits are not suitable for the pattern recognition problems we investigate here, because of limitations of the training algorithms implemented on these chips or because of the limited size of the network that can be trained.

The basic operation performed by a neuron during classi-

fication is a weighted sum, followed by a nonlinear squashing function f , typically a hyperbolic tangent or approximation thereof:

$$y = f(\sum_i x_i w_i + b).$$

We generally refer to the inputs x_i of the neuron as connections and the w_i parameters as weights. Each input is either tied to the output y of another neuron or to an external input. Optionally, a bias b may be added to the weighted sum.

The total number of connections in neural networks for applications such as handwritten character recognition may amount to 10,000 to several hundred thousands.⁸ Networks that solve more general problems, such as recognition of entire words instead of isolated characters, require even larger numbers of connections. The speed requirements of typical applications call for a few tens to several thousands of classifications per second. For each classification, the network must evaluate one multiplication and one addition for every connection, which translates to a few billion multiply-add opera-

***ANNA evaluates dot products of
state and weight vectors and
applies a nonlinear squashing
function to results.***

tions per second. Only parallel implementations, in which several connections are evaluated concurrently, achieve such computational power.

The most general network topology permits connections between any two neurons. Such a high degree of (possible) connectivity, combined with the need for parallel processing, results in enormous hardware requirements, and therefore calls for a compromise. Usually, the neurons in a network are arranged in layers, each of which receives inputs only from neurons in the previous layer. Layers may be fully connected; that is, each neuron may be connected to every neuron in the preceding layer. Often, however, we use local connectivity to express knowledge about the problem (geometric relations such as the neighborhood of pixels in an image) in the network architecture and thus improve the recognition performance.¹

For example, the fact that some pixels in an image are adjacent to each other can be built into the network architecture by constraining neurons to receive inputs only from neigh-

boring pixels. In a fully connected topology, such information must be derived from the training set during the learning phase, usually meeting with only partial success.

A neural network processor could be designed to implement only networks with fully connected topology. Local connectivity would then be realized by simply setting the weights of unused connections to zero. Since, in typical neural networks, the ratio of such unused connections to actual connections is easily 100, such an implementation is unacceptably inefficient. The added complexity of the hardware required to support local connectivity is no match for the millions of connections saved.

Another challenge for a compact hardware implementation of a classifier is the amount of memory needed for storing several tens or hundreds of thousands of weights. Fortunately, the weights of many neurons in important connection topologies, including time-delay or feature extraction neural networks,^{1,2,9} are identical. In these architectures the connection topology corresponds to a one- or higher dimensional convolution, followed by the nonlinear squashing function, as is illustrated. We can realize such a structure with a single, time-multiplexed neuron with a corresponding saving of storage and computing devices.

We can further optimize the hardware complexity by matching the computational accuracy of the processor to the requirements of typical neural networks. Both experience and theory¹⁰ indicate that neural network classifiers can be designed to be insensitive to low-resolution arithmetic. Experiments with character recognizers show that the recognition performance remains virtually unchanged when the inputs and outputs of the neurons are quantized to 3 bits, and the weights to approximately 5 bits. Higher resolution is required in the last layer for the rejection of ambiguous or unclassifiable patterns. Since in typical neural networks the output layer contains only a small fraction of the total number of connections, we reduce system complexity by evaluating those connections on a different processor with higher accuracy.

ANNA

Figure 1 shows the building blocks of ANNA, a neural network chip that implements the concepts just outlined. It concurrently evaluates several dot products of state and weight vectors and applies a nonlinear squashing function to the results. Data enters the chip through a shift register, which reads up to four values at a time. A file with 16 vector registers stores intermediate results when multilayer networks are evaluated.

The 64-word-wide (3 bits per word) shifter reads up to four inputs in each cycle. In this process, the current shifter contents shift left one to four word positions. The use of a shifter limits the number of pins required and supports convolutional network topologies and multiplexing of neurons with identical weights. The shifter alone handles one-dimensional convolutions, while an external data formatter, for example, a line

delay register,¹¹ is needed for two- or higher dimensional computations. Data loaded into the chip can be buffered temporarily in a file of 16 vector registers to reduce the required input bandwidth, to evaluate neurons with more than 64 inputs, or to store intermediate results.

Eight banks of vector multipliers perform the actual computation. Each bank consists of a latch to hold the state vector plus eight vector ALUs with 64 synapses each. A multiplexer that can be configured to combine the contributions from one to four vector multipliers connects the outputs from the vector multipliers to the neuron bodies. When the latches of several vector multiplier banks hold different data, the network evaluates neurons with up to 256 inputs. The number of neurons depends on the number of inputs: Extremes of 16 neurons with 256 inputs each, or 256 neurons with 16 inputs, as well as many intermediate arrangements are possible.¹² The topology can be rearranged on a per-instruction basis to permit evaluation of several layers of a network with different architectures on a single chip without performance penalty.

The neuron bodies first scale the output from the vector multipliers by a factor that can be set in the range 1/16 to 1/2 in eight levels to optimize the useful dynamic range of the circuit. Then the neuron bodies evaluate the squashing function and convert the result to the same 3-bit, signed magnitude representation used at the input of the chip. The weights in the vector multipliers are stored as charge packets on capacitors and must be refreshed periodically. Two on-chip digital/analog converters (DACs) update the values of two different synapses in each clock cycle for a refresh speed of 110 μ s for the entire array.

The chip is programmed with three instructions, CALC, SHIFT, and STORE, to perform computations, load data from an external data source, and transfer data between the shifter, register file, and vector multiplier banks. Parameters for each instruction determine shift count, source of data, number of inputs per neuron, or neuron gain. The CALC instruction executes in four 50-ns cycles, and the network evaluates the other two operations in one clock cycle

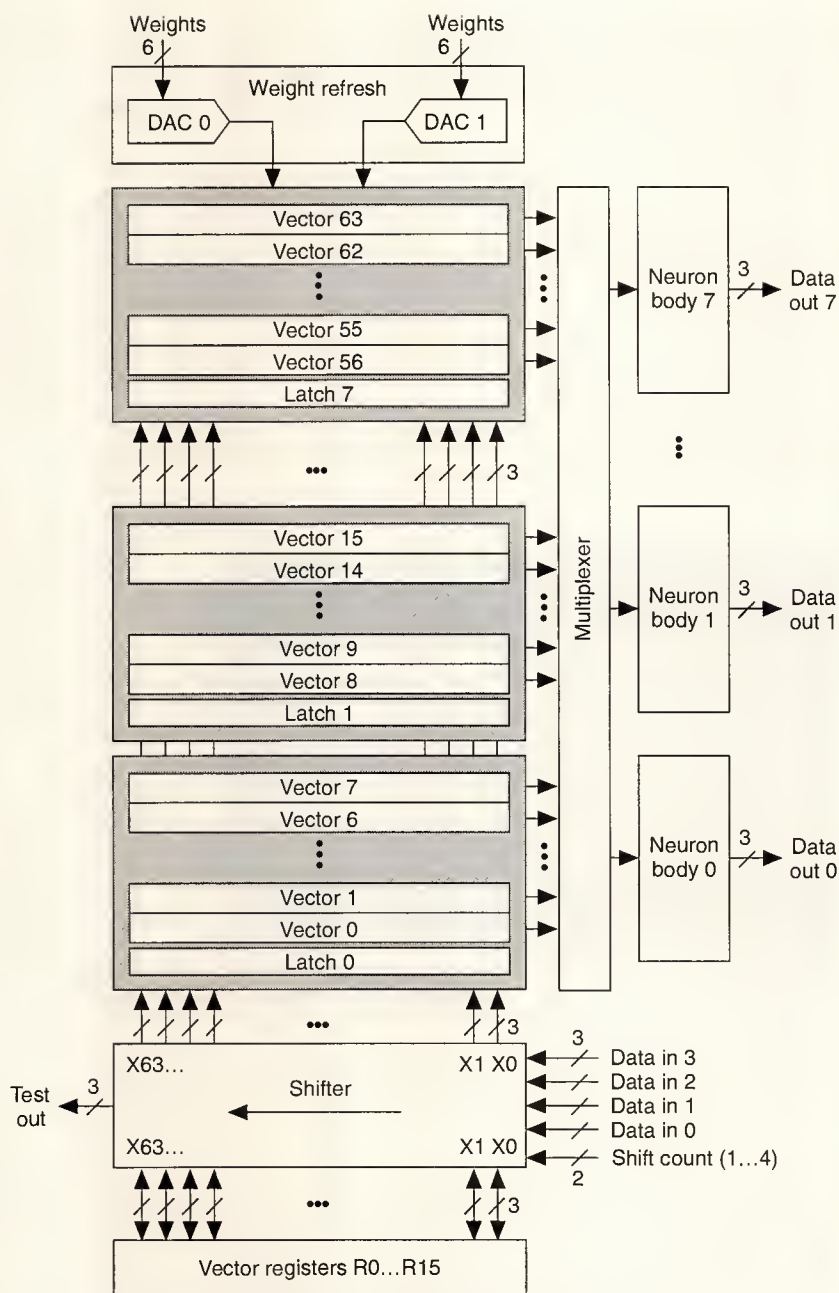


Figure 1. Block diagram of the neural network chip, ANNA.

concurrently with an ongoing CALC instruction. In 200 ns the chip can, for example, load eight states and store them in a register and two latches, and evaluate the dot product and nonlinear function of eight vectors with 256 components each. The weight refresh takes place simultaneously transparent to

Table 1. System features.

Characteristic	Value
Synapses	4,096
Bias units	256
Synapses per neuron	16 to 256
Weight accuracy	6 bits
State accuracy	3 bits
Input rate	120 Mbps
Output rate	120 Mbps
On-chip data buffers	4.6 Kbits
Computation rate (sustained)	5 Gcps
Refresh (all weights)	110 μ s
Clock rate	20 MHz

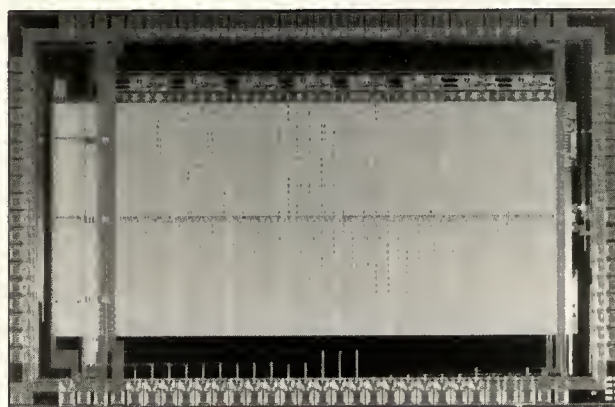


Figure 2. Die photograph. The synapse array can be seen in the center, the shifter and register file on the left, the neuron bodies at the top, and the weight refresh DACs on the right.

the user. Table 1 summarizes the features of the chip.

The chip contains 180,000 transistors and measures 4.5×7 mm² (see Figure 2). It was fabricated in single-polysilicon, double-metal, 0.9- μ m CMOS technology with a 5V power supply. The current drawn by the chip reaches 250 mA when all weights are programmed to their maximum value but is less than 100 mA in typical operation.

Programmability is one of the key features of the neural network chip. Table 2 lists a selection of network topologies that can be implemented and the achieved performance in each case. The chip processes networks with full or sparse connection patterns of selectable size, as well as networks with feedback at a sustained rate of over 10^9 connections per second.

Table 2. Sample network architectures and performance.

Network topology	Average performance (GC/s)
Fully connected (one layer)	
64 inputs, 64 outputs	2.1
128 inputs, 32 outputs	1.2
32 inputs, 128 outputs	1.2
Local receptive fields	
64 \times 1, 64 features	2.3
16 \times 16, 16 features	4.7
16 \times 8, 32 features	3.6
Multilayer network	
64 inputs, 32 hidden, 32 hidden, 32 outputs	0.8
Hopfield neural network	
64 neurons	2.1

Of particular importance for neural network pattern classifiers are neurons with local receptive fields and weight sharing, such as TDNNs.⁹ The neural network chip supports weight sharing in several ways. The shifter and register file enable loading of data and the computation to go on in parallel. Also, data that has been loaded onto the chip once can be buffered and reused in a later computation. Finally, rather than requiring separate hardware for all weights, neurons with identical parameters are stored only once.

Development system

As mentioned before, the characteristics of the ANNA chip—high speed, parallel computation, limited instruction set, and low resolution—differ considerably from those of general-purpose computers. Efficient algorithms that derive optimal benefit from the special processor can be designed only if the processor is available in the early design stages. A development system consisting of an ANNA, a workstation, and appropriate software addresses this requirement. Figures 3 and 4 illustrate the hardware setup, which includes an ANNA and a 20-Mflops DSP32C with 1-Mbyte fast static RAM.

A DMA interface that directly maps the SRAM into the address space of the PC bus or VMEbus exchanges data with the host computer. The DSP, which is also used for pre- and postprocessing and for computations that require higher precision than that of ANNA, generates instructions for ANNA. The entire system is controlled by a program running on the workstation that calls routines and exchanges data with the DSP transparently to the user. The software for the system is written in the high-level language C++, with the exception of a few time-critical routines that are handcoded in DSP assem-

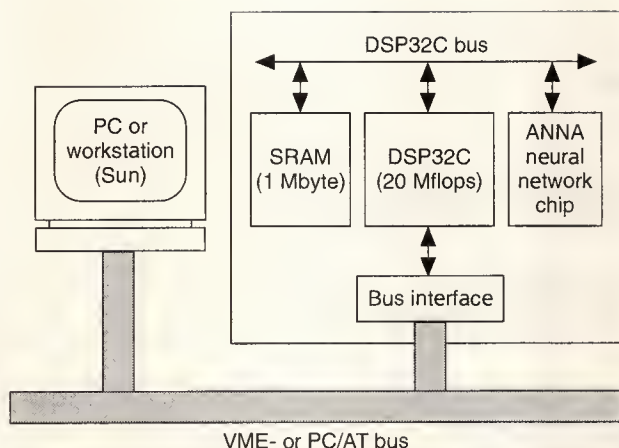


Figure 3. Block diagram of the neural network accelerator board.

bly language.

Networks are trained on the workstation or the DSP. The neural network chip can also be included in the training process, for example, to adjust the network to ANNA's low-resolution processing. Training of individual chips is not necessary, however, because of the good matching between individual devices. Once trained, the network topology and weight values are downloaded into the DSP for execution on ANNA.

Character recognition

Speed, capacity, and programmability are important aspects of neural network hardware. Their practical relevance, however, must be proven on a real-world application, such as the implementation of an optical digit recognizer¹ on the neural network chip we describe here. This network has been trained with the backpropagation algorithm⁴ to recognize handwritten digits from a 20×20 -pixel image. The classification error rate on a test set consisting of 2,000 handwritten digits is 4.9 percent miss classifications, compared to a human performance of 2.5 percent on the same data.

Figure 5 illustrates the architecture of the network; Table 3 lists statistical information about each layer. The more than 3,500 neurons with a total of over 133,000 connections are arranged in five layers. The first four layers employ a 2D convolutional topology with various kernel sizes and subsampling factors. Because of weight sharing, the number of weights (free parameters) in these layers is much smaller than the number of connections. The last layer is fully connected. We chose this topology to maximize recognition performance and classification speed of an implementation on a floating-point DSP32C digital signal processor.¹³

Special steps are necessary to adapt the network to the low resolution of the chip. Simple quantization of all weight values

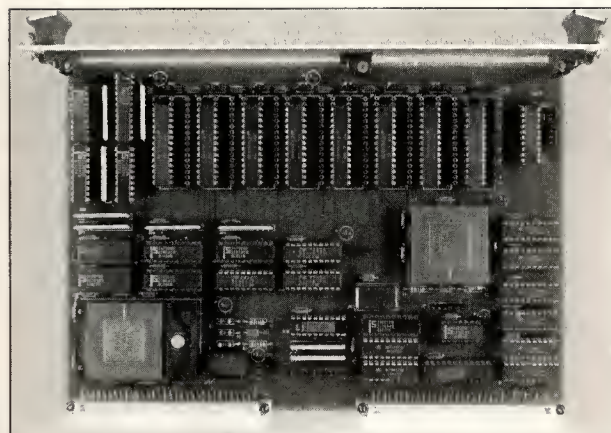


Figure 4. Neural network accelerator with ANNA and the DSP32C.

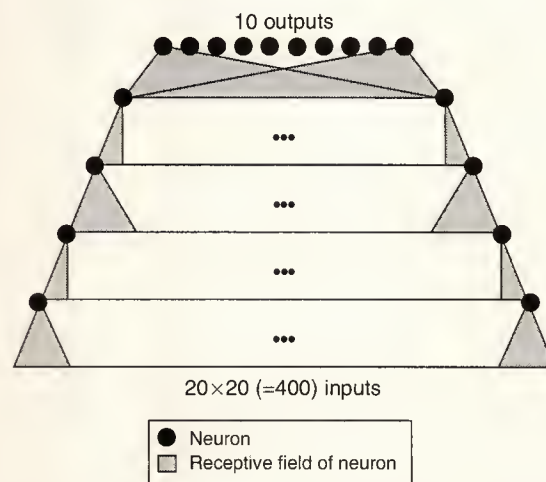


Figure 5. Architecture of the character recognition network.

Table 3. Connectivity of character recognizer neural network.

Layer	Neurons	C/s	Weights
5	10	3,000	3,000
4	300	1,200	12
3	1,200	50,000	500
2	784	3,136	4
1	3,136	78,400	100

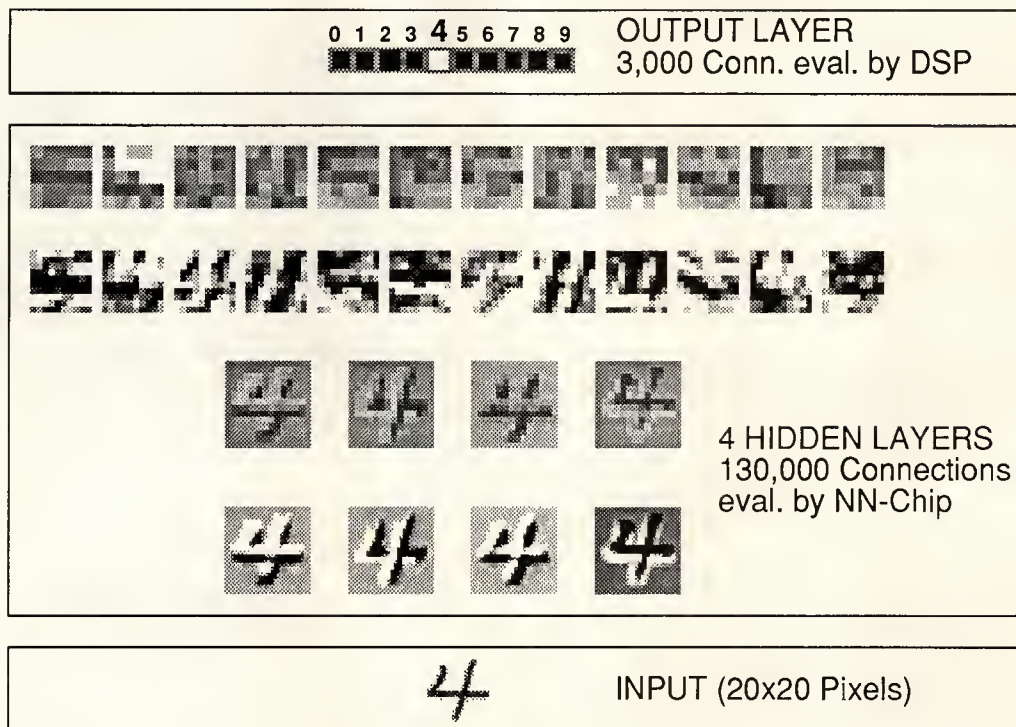


Figure 6. Sample chip output for optical character recognition. The gray levels encode the neuron state.

results in an unacceptable loss of accuracy. However, experiments reveal that the computational accuracy provided by the chip is adequate for all but the 3,000 weights in the last layer of the network. This last layer is retrained with quantized data obtained from the chip to eliminate performance degradation. After retraining, the classification error rate on the test set is 5.3 percent, compared to the original 4.9 percent. This result is obtained consistently with different chips for which the last layer has not been retrained individually. Figure 6 shows the input, output, and internal states of the neural network for a sample input that has been processed by the neural network chip.

The first four layers of the network with 97 percent of the connections but only 616 weights fit on a single neural network chip. The remaining 3,000 connections of the last layer are evaluated on the DSP32C. The throughput of the chip is more than 1,000 characters per second or 130,000 connections per second. This figure is considerably lower than the peak performance of the chip (5G connections per second), a consequence of the small number of inputs of most neurons in the network for which the chip cannot fully exploit its parallelism. Nevertheless, the chip's performance compares favorably to the 20 characters per second that are achieved when the

entire network is evaluated on the DSP32C. The recognition rate of the chip is far higher than the throughput of the preprocessor, which relies on conventional hardware. Improvements of both the recognition rate and accuracy can be expected when the network architecture is tuned to take full advantage of the parallelism of the ANNA chip.

NEURAL NETWORKS ARE ATTRACTIVE FOR PATTERN classification applications but suffer in practice from the limited speed that can be achieved with implementations based on classical processors. This problem can be overcome with highly parallel special-purpose VLSI circuits. While a fully parallel implementation of sufficiently large networks is currently not feasible, we can achieve adequately high performance with an architecture that exploits the limited connectivity and weight sharing that are typical for pattern classifiers. We demonstrated this performance with a neural network classifier with over 133,000 connections that has been implemented on a single

neural network chip performing over 1,000 classifications per second. This result eliminates throughput from the constraints faced by network designers. The availability of fast special-purpose hardware for large applications sets the conditions to explore new neural network algorithms and problems of a scale that would not be feasible with conventional processors.

We expect further advances when the architecture of the network is modified to fully take advantage of the chip's parallelism. While the size of the current network has been constrained by the speed of conventional hardware, such issues vanish because of the high speed of the chip. The price for this throughput is the specialization of the circuit, specifically its low resolution, and its focus on neural network algorithms. Future research will benefit from the speed of the novel hardware but must also address questions regarding the limitations of special-purpose hardware. Furthermore, it appears attractive to implement larger tasks (to include image location, segmentation, and scaling into the recognition process) with neural networks to benefit from the powerful hardware. ■

References

1. Y. leCun et al., "Handwritten Digit Recognition with a Back-Propagation Network," in *Neural Information Processing Systems*, Vol. 2, D.S. Touretzky, ed., Morgan Kaufmann Publishers, San Mateo, Calif., 1990.
2. I. Guyon et al., "Design of a Neural Network Character Recognizer for a Touch Terminal," *Pattern Recognition*, Vol. 24, No. 2, 1991, pp. 105-119.
3. B.E. Boser and E. Sackinger, "An Analog Neural Network Processor with Programmable Network Topology," *ISSCC Dig. Tech. Papers, Proc. IEEE Int'l Solid-State Circuits Conf.*, Vol. 34, Feb. 1991, pp. 184-185.
4. D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing—Explorations in the Microstructure of Cognition*, Vol. 1, MIT Press, Cambridge, Mass., 1986.
5. M. Mexard and J.P. Nadal, "Learning in Feed-Forward Layered Networks: The Tiling Algorithm," *J. Phys.*, Vol. A-22, 1989, pp. 2191-2203.
6. Y. Arima et al., "A 336-Neuron, 28K-Synapse, Self-Learning Neural Network Chip with Branch-Neuron Architecture," *ISSCC Dig. Tech. Papers, Proc. IEEE Int'l Solid-State Circuits Conf.*, 1991, pp. 182-183.
7. J. Alspector, R. Allen, and A. Jayakumar, "Relaxation Networks for Large Supervised Learning Problems," *Neural Information Processing Systems*, Vol. 3, R. Lippmann, J. Moody, and D. Touretzky, eds., Morgan Kaufmann Publishers, 1991, pp. 1015-1021.
8. L.D. Jackel et al., "VLSI Implementations of Electronic Neural Networks: An Example in Character Recognition," *Proc. IEEE Int'l Conf. Systems, Man, and Cybernetics*, 1990, pp. 320-322.
9. A. Waibel et al., "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Vol. 37, No. 3, Mar. 1989, pp. 328-339.
10. M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of Feedforward Neural Networks to Weight Errors," *IEEE Trans. Neural Networks*, Vol. 1, No. 1, Mar. 1990, pp. 71-80.
11. P.A. Ruetz, "The Architectures and Design of a 20-MHz Real-Time DSP Chip Set," *IEEE J. Solid-State Circuits*, Vol. 24, No. 2, Apr. 1989, pp. 338-348.
12. H.P. Graf and D. Henderson, "A Reconfigurable CMOS Neural Network," *ISSCC Dig. Tech. Papers, Proc. IEEE Int'l Solid-State Circuits Conf.*, Vol. 33, 1990.
13. M.L. Fuccio et al., "The DSP32C: AT&T's Second-Generation Floating-Point Digital Signal Processor," *IEEE Micro*, Vol. 8, No. 6, Dec. 1988, pp. 30-48.



Bernhard E. Boser is an assistant professor in electrical engineering at the University of California, Berkeley. While writing this article, he was working on algorithms and parallel hardware for artificial neural networks at AT&T Bell Laboratories.

Boser received a diploma in electrical engineering from the Swiss Federal Institute of Technology in Zurich, Switzerland, and MS and PhD degrees from Stanford University. He is a member of the IEEE and the Computer Society.



Eduard Sackinger is a member of the technical staff at AT&T Bell Laboratories in Holmdel, New Jersey, on artificial neural-network hardware and its applications to character recognition. He previously worked at the Electronics Laboratory of the Swiss Federal Institute of Technology

where he investigated analog applications to floating-gate devices. His research interests include analog circuit design and parallel distributed processing.

Sackinger received the MS and PhD degrees in electrical engineering from the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. He is a member of the IEEE.

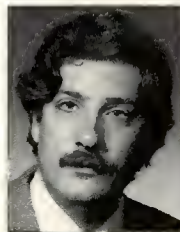


Jane Bromley, a consultant at AT&T Bell Laboratories, works on the application of artificial neural networks to human perception tasks, including the reading of handwritten text. She received the BSc degree in physics and the PhD degree in biophysics from Imperial College, London University. She is a member of the American Psychological Society and the Association for Research in Vision and Ophthalmology.



Yann leCun is a member of the technical staff at AT&T Bell Laboratories. He previously worked as a research associate in the Department of Computer Science of the University of Toronto, Canada. His current interests include neural networks and connectionist models, learning theory, pattern recognition, and VLSI implementations of massively parallel systems.

LeCun obtained an engineering diploma in electrical engineering from the School of Electronic and Electrotechnic Engineering, Paris. He holds a PhD degree in computer science from the Piere and Marie Curie University, Paris, during which time he introduced an early version of the backpropagation algorithm. He served as session chair on the organizing committees of several conferences, including the International Joint Conference on Neural Networks, the Neural Information Processing Systems Conference, and the International Neural Network Conference.



Lawrence D. Jackel heads the Adaptive Systems Research Department at AT&T Bell Laboratories in Holmdel, New Jersey, where he pursues research into theory, applications, and hardware for machine learning and machine perception. His initial research was in microfabrication and

microscience.

Jackel received a BS degree in physics from Brandeis University in Waltham, Massachusetts. He was awarded the PhD degree in experimental physics from Cornell University in Ithaca, New York. He is a member of the IEEE, the Computer Society, and the American Physical Society.

Direct questions concerning this article to Bernhard E. Boser, EECS Dept., Cory Hall, University of California, Berkeley, CA 94702.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

High 156

Medium 157

Low 158



1992 Gordon Bell Prize

For Outstanding Achievement in the Application of Parallel Processing to Scientific and Engineering Problems

Entries are due May 1, 1992, with finalists to be announced by June 30 and winners announced at the Supercomputing 92 conference in November 1992. Prizes of \$1,000 each will be awarded in two of three categories: performance, price/performance, and compiler parallelization.

For more information, contact Claire Azada, IEEE Computer Society, (714) 821-8380.

Editorial Calendar

APRIL 1992

Hot Chips III

- This extremely popular issue presents the latest developments in microprocessor and chip technology used to construct high-performance workstations and systems as presented at the annual IEEE Computer Society-sponsored symposium
- Past contributors include: Intel, Motorola, Apple, Sun, IBM Research, Metaflow Technologies, Intergraph, ITT

Ad closing date: March 1

OCTOBER 1992

Processing hardware for video communication

- ICs for HDTV compression
- ICs for HDTV communication
- Parallel processing systems with real-time video compression capabilities
- Multimedia systems with real-time video compression capabilities

Ad closing date: September 1

JUNE 1992

Associative memories and processors

- Late-breaking developments in wide-word content-addressed memories (CAMs)
- Dynamic CAPP (parallel processor) architectures
- Fault-tolerant architectures for crucial control systems such as those in trains, space systems, etc.

Ad closing date: May 1

DECEMBER 1992

Special Signal Processors

- Recent information from the vital area of digital signal processors
- News about other techniques for signal processing
- Mixed analog/digital processors solve a real need
- Neural networks process signals following methods used by the human brain

Ad closing date: November 1

AUGUST 1992

European industry

- Recent developments in integrated circuit and microsystem technology from major European manufacturers

Ad closing date: July 1

FEBRUARY 1993

Automotive/traffic microelectronics

- Worldwide developments in microelectronics for traffic and driving assistance
- Latest developments in the European Prometheus and US IVHS programs
- News from Japan, too
- Improving traffic safety with electronics

Ad closing date: January 2

IEEE Micro helps designers and users of microprocessor and microcomputer systems explore the latest technologies to achieve business and research objectives. Feature articles in IEEE Micro reflect original works relating to the design, performance, or application of microprocessors and microcomputers. All manuscripts are subject to a peer-review process consistent with professional-level technical publications. IEEE Micro is a bimonthly publication of the IEEE Computer Society.

Advertising information: Contact Marian Tibayan, Advertising Department, IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax (714) 821-4010.

Articles may change. Please contact editor to confirm.



Experimentation with Hypercube Database Engines

Effective algorithms appropriately selected for each task are essential if we are to take full advantage of parallelism in database systems. Some algorithms perform faster than others, depending on the uniqueness value of the data and the uniformity of distribution among the nodes in the system. A performance evaluation under varying experimental parameters determines the optimality of a given algorithm for a particular database task.

Ophir Frieder

Vijaykumar A. Topkar

Ramesh K. Karne

Arun K. Sood

George Mason University

Interest in multiprocessors for database processing has grown beyond the research communities, into the development sectors. As the volume and variety of data stored, accessed, and manipulated grows, computer and communications designers are focusing on parallelism—in particular using general-purpose commercial multicomputers—as a way to meet database processing needs.¹⁻³

Using one such computer, Intel's iPSC/2 hypercube, we measured the relationship between packet size, method of clustering messages, and internode traffic on the total sustained communication bandwidth. Having measured the costs associated with internode communication, we then analyzed duplicate removal algorithms. Duplicate removal is an integral part of several frequently used relational database operations, including PROJECT and UNION. We believe it is, therefore, important to develop efficient algorithms.

We also studied the effects of nonuniformly distributed attribute values and tuples across processors on three proposed duplicate removal al-

gorithms. We chose algorithms to represent the several available in the literature.⁴⁻⁷ We then evaluated the output collection time.

A tutorial on multiprocessor/multicomputing databases⁶ or current generation multicomputing architectures⁸ is beyond the scope of this article. We intend only to present a brief overview of the iPSC/2's hypercube message-passing system, and discuss the results of our experimentation and analysis. Although our algorithms were implemented on the iPSC/2, we believe they would work as well on other hypercube machines, such as Ncube⁹ and Floating Point Systems' T series.¹⁰

iPSC/2

Intel's Personal Super Computer, the iPSC/2, is a multiple-instruction, multiple-data (MIMD), multicomputer with nodes connected via a hypercube topology. Although many definitions for a multicomputer system exist, we define it as a system comprised of multiple nodes, each of which is a complete computer. That is, each node has a set of resources (dedicated I/O, memory, and CPU) and is under the independent control

of its own operating system.

Message-passing application interface. Application programs access the iPSC/2's message-passing system through system calls.¹¹⁻¹³ The NX/2 operating system provides three levels of system calls: synchronous, asynchronous, and interrupt-driven. Users can mix and match these three types of calls. For example, a message sent with an asynchronous call can be received either with a synchronous call or an interrupt-driven call. The size of the message is limited only by the amount of physical memory available at the node.

The processor uses an application-development message-passing paradigm. Consider, for example, a node program that receives data from the host or from another node and then locally processes them. If the node cannot locally process without the received data, then it must use a synchronous receive call to receive the data.

On the other hand, if the node can accomplish some of the local processing without the received data, then it uses an asynchronous receive. After initiating the asynchronous receive, the node carries out the local processing that does not require the received data. The node program then checks if the data have been received. If the data have not been received, the program waits.

The synchronous calls are also called blocking calls. When a node sends a message using this protocol, it is blocked for processing until the operating system copies the send buffer into its local buffer and is ready to send this message to its destination. This operation does not imply however that the message sent is received at the destination node. The message buffering and flow control in NX/2 ensure the synchronous transmission intended by the application.

The asynchronous calls are nonblocking calls. When a node sends a message using this protocol, the sending node is free to process soon after the execution of that call. The receiving node must poll for any messages received asynchronously. This protocol facilitates the application to perform other tasks while the messages are in transit.

Message-passing measurements. We conducted the following experiments to analyze the communication behavior of iPSC/2's message-passing system. We measured the communication times between two adjacent nodes that are also at the extreme diagonals on the hypercube. The communication times for a one-hop (adjacent nodes) and multihop (nonadjacent nodes) are almost equal, because the intermediate nodes take only a few microseconds to help establish a path. Thus, their computation phase is uninterrupted. This indicates that the iPSC/2 message-passing interconnection, though organized as a hypercube topology, practically functions as a fully connected interconnection network.

A fully connected 128-node system has eight simultaneously available paths, that is, paths without contention. However, if multiple nodes simultaneously send messages to a common destination, then contention is imminent. To evaluate

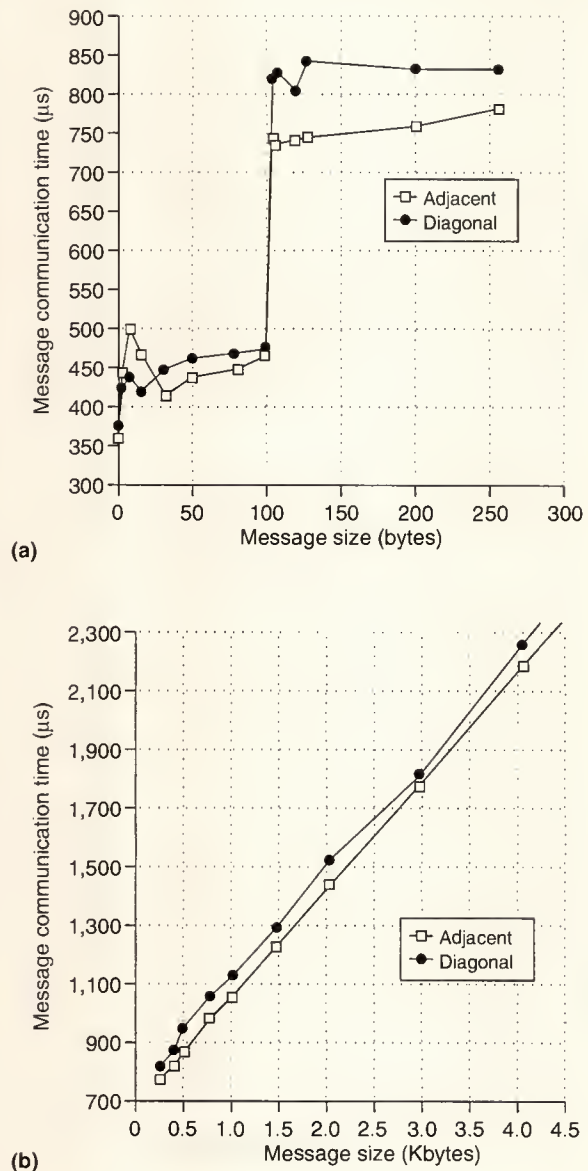


Figure 1. Communication time as a function of message size: small (a), large (b).

the effects of contention we measured the communication times when one or more source nodes send messages to a single receiving node. The communication times at the source nodes slow down proportionally to the number of source nodes increases.

In our experiments, we used an eight-node system and measured average communication time of 1,000 messages in which each node sends to and receives from another node. Figure 1 shows the communication time for varying message

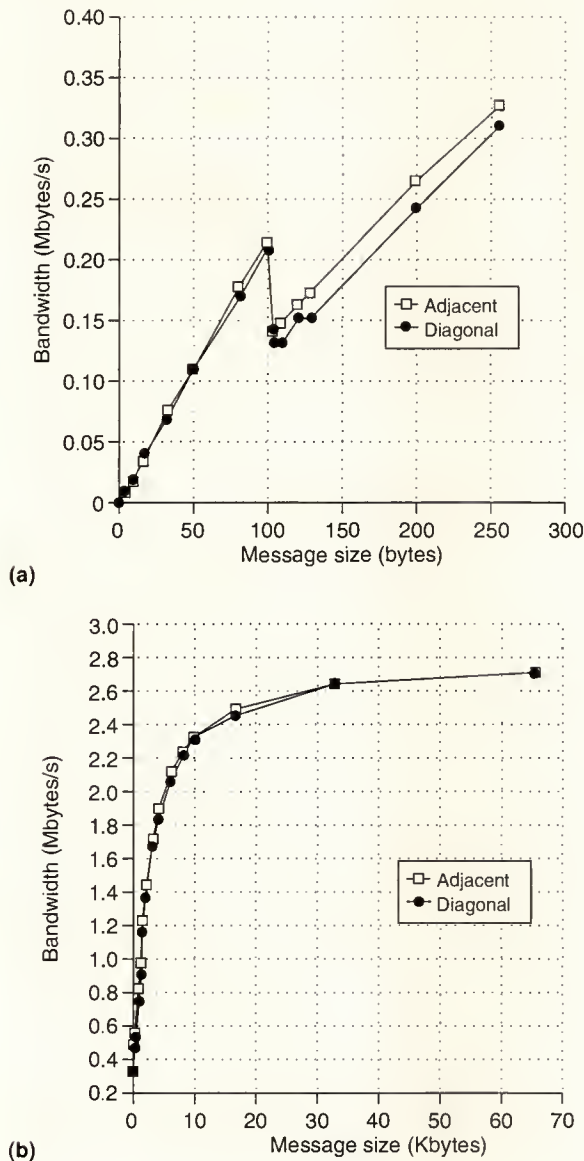


Figure 2. Communication bandwidth as a function of message size: small (a), large (b).

sizes between neighbor nodes and nodes three hops apart. Figure 2 illustrates the computed bandwidths of the communication channel for different operating conditions. Note that the communication time for longer messages is comparable to that of smaller messages. Consequently, we prefer implementations requiring fewer but longer messages.

We measured the communication time for a zero-byte mes-

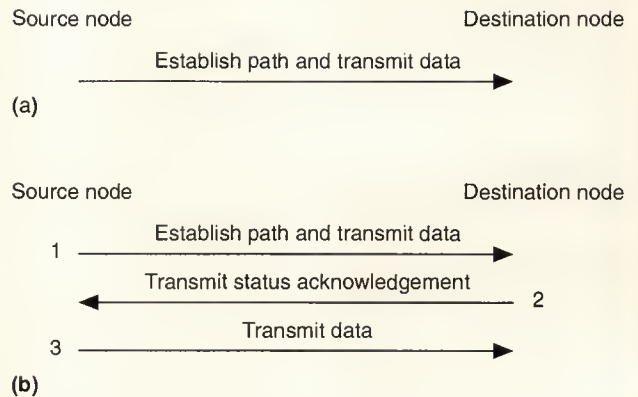


Figure 3. Message transmission protocols: one-step (a), three-step (b).

sage at 361 μ s. This is the minimum setup time required to transmit a message of any size. In addition to the setup time, messages larger than zero bytes need a propagation time.

The system uses two protocols for message transmission. For short messages (zero to 100 bytes), a one-step protocol, shown in Figure 3a, sets up a path and transmits data concurrently.

Figure 1a shows that, for messages larger than 100 bytes, the communication time increases by almost 280 μ s. (Since all messages include a 4-byte boundary, a 100-byte message actually takes 104 bytes.) The time increases because the system uses a three-step protocol, shown in Figure 3b, for larger messages. In the first cycle, the source node establishes a path by sending a status message to the destination node. In the second, if the destination node is ready to receive a large message, it sends back a status acknowledgment. In the final step, the source node sends the complete message to the destination node.

This three-step protocol increases the communication time for large messages, but two different protocols are required since the one-step protocol could overflow the memory buffer if used for large messages. Alternatively, the three-step method would add unnecessary protocol overhead if used for small messages.

Figure 1b shows the communication time chart for messages between 256 and 4,096 bytes. Note that communication time for extreme diagonal nodes and adjacent nodes is almost equal. Figure 2a shows the bandwidth chart for messages from 0 to 256 bytes. Figure 2b shows the bandwidth chart for messages in the range from 256 bytes through 64 Kbytes. To achieve a 2.7-Mbyte/s bandwidth, the message size must be at least 64 Kbytes. For small messages, say 100 bytes, the bandwidth is 0.21 Mbytes/s.

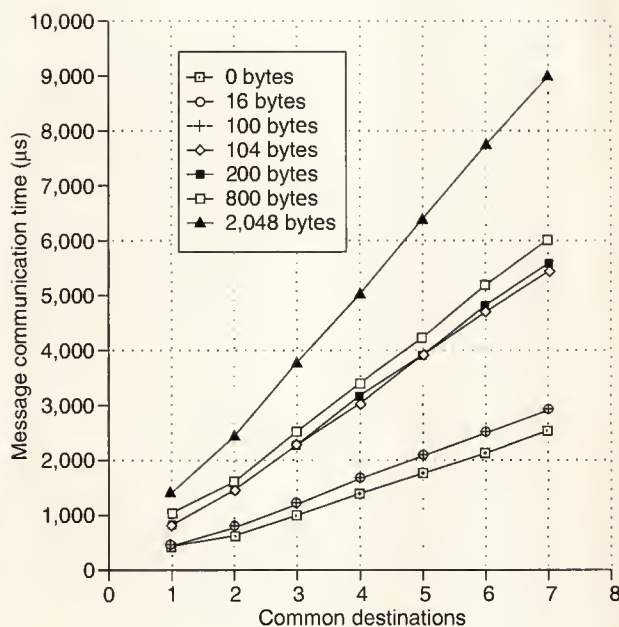


Figure 4. Number of common destinations versus time.

We also measured communication times for message transmissions with contention in the network. Contention is created by transmitting messages from one or more source nodes to a common destination node. To demonstrate, we chose an eight-node hypercube with an arbitrarily chosen destination node 5. Nodes 0 through 4, 6, and 7 transmit messages to node 5 and the transmission times are recorded.

Figure 4 shows the number of common destinations versus message communication time. Each curve represents a different message size. The message communication time increases almost linearly, as the number of common destinations increase. For example, sending a 100-byte message from a single source node takes 501 μ s, while for seven source nodes, the communication time is 3,006 μ s. For every 100 bytes of memory, the communication time increases approximately 400 μ s for each common destination node added. Similarly, for a 104-byte message, the communication time increases approximately 800 μ s for each common destination node. (The dramatic increase for a 104-byte message is due to the three-step protocol.)

We also observed that the communication time increase caused by contention is independent of the location of the common destination node. This increase stems from the communications technology employed in the iPSC/2. These measurements suggest that, to achieve scalable performance, we should use algorithms that avoid network contention.

Our experiment led us to the following conclusions:

- *Designers should choose an appropriate message size based on the requirements of an application.* The three-step protocol causes extra overhead in the transmission of messages. Thus, if the message size is only slightly over 100 bytes, it may be advantageous to reduce the message size.
- *For short message transmissions, the iPSC/2 offers a quite low bandwidth, on the order of Kbytes.* Thus, message clustering may be required to achieve higher performance. In message clustering, one or more messages destined for the same target node are collected in a packet and transmitted as a single message packet. However, message clustering increases processing overhead. For some applications, this increased overhead may nullify the benefits of using the high bandwidth.
- *The iPSC/2 effectively supports a fully connected network.* With the notable exception of minimizing node contention, there are few additional benefits to partitioning the application to suit the adjacent node communications.
- *Simultaneous data transfer to the same node dramatically increases the communication time.* Consequently, we prefer algorithms that avoid such contention.

We designed the algorithms presented in the following sections with these communication constraints in mind.

Relational database

Numerous commercial relational database systems are available that execute on a range of host processing systems including personal, mini- and mainframe computers, as well as a host of multicomputer environments. Each system's performance depends on the underlying execution environment and the liberties taken by the software developer. For example, duplicate tuples (or rows) are not allowed in the actual relational database model,¹⁴ and yet several vendors support duplicate tuples. We initially describe a subset of relational operators and then provide a discussion of three duplicate removal algorithms that can be employed as part of the implementation of these operators.

Operators. The relational database model is the underlying environment for a wide diversity of applications. For example, medical pictorial databases, commonly called picture archiving and communication systems (PACS), employ the relational model to store and access patient data.¹⁵ Some designers propose protocol verification systems that use the relational model to implement an efficient algorithm based on a reachability analysis technique.¹⁶ Thus, the development of high-performance parallel algorithms for the relational operators impacts not only the traditional consumer database processing arena but also nontraditional database domains, including the communications and medical communities.

Table 1. Relations *r* and *s*.

Relation <i>r</i>		Relation <i>s</i>		
Employee number	Age	Employee number	Children	Gender
0	33	0	3	M
1	25	1	0	M
2	53	5	3	F
3	25	7	2	F
4	25	8	1	F
5	45	10	4	M
6	28	16	8	F
7	45	17	4	F
8	64	18	2	M
		22	1	F
		28	0	M
		31	0	F
		42	3	F
		43	2	M

To review relational database nomenclature, consider the relational database shown in Table 1. This database consists of two relations, *r* and *s*. Relation *r* has two attributes: employee number and age. Relation *s* has three attributes: employee number, number of children, and gender.

Each attribute is defined over some finite or countably infinite domain. In this example, the attributes in *r* are defined over the set of whole numbers and the set of whole numbers between 0 and 120. The attributes in *s* are defined over the domains of the set of whole numbers, the set of whole numbers between 0 and 30, and the set {M, F}. Relation *r* consists of nine tuples; each tuple contains two attributes:

{<0, 33>, <1, 25>, <2, 64>, <3, 25>, <4, 25>, <5, 53>, <6, 28>, <7, 45>, <8, 64>}.

Similarly, *s* consists of 14 tuples, each comprised of three attributes.

Several popular operators exist for the manipulation of the relations comprising the database. Here, we briefly overview only four: Project, Union, Select, and Join. Formally specified, these operators are defined as follows.

- *Project*. The projection on $P[XYZ]$, denoted as $\pi_A(P)$, is defined by $\pi_A(P) = \{x[A] \mid x \in P\}$, where *A* is a set of attributes of *P*.
- *Union*. The union of two relations, $P[XYZ]$ and $Q[XYZ]$, denoted as $P[XYZ] \cup Q[XYZ]$, is defined by $P[XYZ] \cup Q[XYZ] = \{x \mid x \in P \text{ or } x \in Q\}$, where *X*, *Y*, and *Z* are

a disjoint set of attributes.

- *Select*. The selection on $P[XYZ]$, denoted as $\sigma_{B\oplus b}(P)$ where $\oplus \in \{<, \leq, =, \geq, >, \neq\}$ is defined by $\sigma_{B\oplus b}(P) = \{x \mid x[B] = b, x \in P\}$, where *B* is an attribute of *P*.
- *Join*. The join of two relations $P[XYZ]$ and $Q[VWX]$, denoted as $P[XYZ] \bowtie Q[VWX]$, is defined by $P[XYZ] \bowtie Q[VWX] = \{u \mid p \in P, q \in Q, p[X] = q[X], u[XYZ] = p, u[VWX] = q\}$, where *V*, *W*, *X*, *Y*, and *Z* are a disjoint set of attributes. If no common joining attributes exist, the join of *P* and *Q* is the Cartesian product of *P* and *Q*.

Given relations *r* and *s* as shown in Table 1, the user requests, or queries, are evaluated in Figure 5.

In typical database applications, most attributes contain duplicate values. Duplicates occur frequently in database processing due to the inherent nature of the data being processed. For example, George Mason University has about 20,000 students in 43 undergraduate programs. Therefore, the program discipline attribute in the student database contains a high degree of duplicates, that is, a low uniqueness factor.

1. Find all employees with three children.

$$\sigma_{\text{Children}=3}(s) =$$

Employee number	Children	Gender
0	3	M
5	3	F
42	3	F

2. Find all employees that are childless or are 25 years old.

$$\pi_{\text{Emp No.}}(\sigma_{\text{Children}=0}(s)) \cup \pi_{\text{Emp No.}}(\sigma_{\text{Age}=25}(r))$$

$$\text{Employee Number}$$

1
3
4
28
31

3. Determine all the available information about 45-year-old women.

$$\sigma_{\text{Age}=45}(r) \bowtie \sigma_{\text{Gender}=F}(s)$$

Employee number	Age	Children	Gender
5	45	3	F
7	45	2	F

Figure 5. Queries.


```

If an attribute value in the local database is also present in the received packet, then
{
    if (local_count > packet_count) then
        local_global_count = local_global_count + packet_count
    else if (local_count < packet_count) then
        delete the tuple from the local database
    else if (local_count = packet_count) then
        {
            if (local_node_address > packet_node_address) then
                local_global_count = local_global_count + packet_count
            else
                delete the tuple from the local database
        }
    }
}

```

Figure 6. Multiple-message ring algorithm (local processing).

In the database presented in Figure 5, duplicate entries appear in the number of children and employee gender attributes of *s*, and in the age of the employees attribute in *r*. In fact, in most cases, only the key attributes are typically unique. We focused on designing algorithms that remove duplicates.

We also studied the effects of low uniqueness on the performance of proposed parallel algorithms. Other authors have written on the computational complexity and processing requirements of relational database queries in the presence of duplicate attribute values.^{7,17,18}

Of the relational database operators presented earlier, we emphasize Project and Union since both can result in duplicate tuple values in the initial stage of processing. That is, the Project operator initially eliminates columns which may result in duplicate tuple values. Thus the removal of duplicates from the generated set of tuples must follow.

The Union operator eliminates the duplicate copies that result from the initial processing stage. In the Union operator, the initial processing stage involves the combining of the tuples of the first relation with the tuples of the second relation. Duplicate removal comprises the second stage. (In a distributed-memory architecture, the first stage can occur without any internode communication, whereas, in the second stage, all nodes must communicate.) The duplicate tuples belong to the intersection of the input relations to the Union operation.

Algorithms for duplicate removal. We describe three parallel duplicate removal algorithms, the multiple message ring (or, simply, the ring), recursive reduction, and bucket algorithms. Each algorithm removes the duplicates in two steps: locally at each node, and then globally throughout the hypercube.

All three algorithms commence with the local duplicate removal phase. This phase forms a local database of tuples with unique attribute values originally available at the given node (that is, no two tuples have the same attribute value). For each

unique datum value, two fields are maintained. One is a local repetition count representing the number of copies of the datum initially present at the node. The other is a global repetition count that is initialized to the corresponding local repetition count before the global duplicate removal stage.

Since the local duplicate removal phase is identical in all three algorithms, we will not elaborate on it. We will briefly describe the global duplicate removal phase for each algorithm, though further details of the algorithms, including analytical

models and performance evaluation, are beyond the scope of this article.⁷

Multiple-message ring. After the local duplicate removal phase, a unidirectional ring is embedded using Reflexive Gray Codes in the $\log_2 N$ -dimensional hypercube. A packet consisting of tuples with locally unique attribute values and their local repetition counts is formed at each node. The global duplicate removal is achieved in $N-1$ iterations. During each iteration, each node sends a packet to the next node, in a pipeline ordering of the Reflexive Gray Code, and receives one from the previous node. The received packet updates the local database as shown in Figure 6. This procedure repeats for $N-1$ iterations, after which the following are true:

- Attribute values are unique across node boundaries.
- The node on which an attribute value is found is the node which had the largest local repetition count to start with. If more than one node had the largest local repetition count to start with, then the node with the largest node address will retain that unique value.
- The final local_global_count of an item is the global repetition count of that value.

Recursive reduction. The global duplicate removal phase is achieved by a $\log_2 N$ -fold recursive reduction of the size of the cube containing the data, with the final results placed at a single target node. In each step *j*, the size of the active cube is halved based on the value of the *j*th bit in the node address. That is, all nodes with the *j*th bit equal to one belong to the first half while the remaining nodes, those nodes with the *j*th bit equal to zero, belong to the second half.

For each node in one half, a corresponding node exists in the other half, where every bit except *j* is identical in value. Nodes that differ from the target node in the *j*th bit transmit local data to their corresponding node. The receiving nodes

```

If a datum in the packet is also present in the local database then
    local_global_count = local_global_count + packet_count
else
    add the datum value to the local database at the appropriate location.

```

Figure 7. Recursive reduction algorithm (local processing).

```

Let N = Number of nodes,  $k = \log_2 N$ , Node address =  $\langle \langle n_{k-1} \ n_{k-2} \ \dots \ n_0 \rangle \rangle$ 
Let max_value and min_value define the range of attribute values to be sorted.
Let recursion_no = 0;
while (recursion_no < k)
{
    create lower bucket consisting of attribute values such that
        attribute value  $\leq \frac{(\text{Maximum attribute value} - \text{Minimum attribute value})}{2}$ 

    create upper bucket consisting of remaining attribute values.
    Let X =  $\langle \langle 00 \ \dots \ 010 \ \dots \ 0 \rangle \rangle$  where the 1 is in the (recursion_no)th position.
    Let partner_node =  $\langle \langle n_{k-1} \ n_{k-2} \ \dots \ n_0 \rangle \rangle \text{ BIT\_WISE\_EX\_OR } X$ 
    if ( $n_{\text{recursion\_no}} = 0$ )
    {
        send upper bucket to the partner_node
        receive bucket from partner_node and merge it with the lower bucket
         $\text{max\_value} = \text{min\_value} + \frac{\text{max\_value} - \text{min\_value}}{2}$ 
    }
    else
    {
        send lower bucket to the partner_node
        receive bucket from partner_node and merge it with the upper bucket
         $\text{min\_value} = \text{min\_value} + \frac{\text{max\_value} - \text{min\_value}}{2}$ 
    }
    recursion_no = recursion_no + 1
}

```

Figure 8. Pseudocode of bucket algorithm.

merge their respective local databases with the packets received, as shown in Figure 7. On termination, all the unique values and their global counts reside in the target node.

Bucket. This algorithm is a special case of the hash join algorithm¹⁹ and can be used whenever the range of attribute values in the system is known a priori or can be readily and efficiently computed. A bucket consists of attribute values within a specified range. Each node is mapped to a bucket, and the attribute values residing on all other nodes belonging to that bucket are routed to that node. The routing is done in $\log_2 N$ steps by following an algorithm similar to the recursive reduction algorithm. During each step j , corresponding nodes partition the attribute values range into two halves. Nodes

whose j th address bit equals zero keep all tuples whose attribute values map to the lower attribute value range and transmit the remaining tuples to their corresponding node. Nodes whose j th address bit equals one keep all tuples whose attribute values map to the upper attribute value range and transmit the remaining tuples to their corresponding node. Figure 8 outlines the pseudocode description of the bucket algorithm.

Communication requirements.

We designed these three algorithms to exploit our knowledge of the iPSC/2's message-passing system and to prevent simultaneous transmission to a node. In general, the internode communication needs of the algorithms are restricted to adjacent nodes.

Unfortunately, the volume of data transferred between nodes is data dependent. Hence, it is difficult to maximize the sustained communication bandwidth between nodes by using large packets. When multiple packets are required to transfer the data between the nodes however, we prefer fewer large packets to many small packets.

Performance evaluation

The time required for duplicate removal indicates the efficiency of the algorithm. The total time consists of three components:

- local duplicate removal,
- global duplicate removal, and
- data collection.

The ring and bucket algorithms distribute the resulting data across all the nodes. If the results must reside at a single node, they must be transmitted. The time needed to accomplish this is called collection.

The inclusion of collection time and the duplicate removal phase, and the proper selection of the algorithm is application dependent. For example, if the starting data are already locally unique, the local duplicate processing time need not be included. Similarly, if the final results are to be used as an input for further processing, the collection time should not be included in the execution time. Unless otherwise stated, we assume that the local duplicate removal time is included

and the collection time is not included for computing the total execution time of an algorithm.

We carried out experiments by varying the work load characteristics and the system configuration, namely the number of nodes. The cardinality, uniqueness, tuple size, distribution of the unique values, and distribution of data among the nodes define the work load characteristics. We define the uniqueness factor (or simply the uniqueness) of the data as the number of distinct attribute values expressed in a fraction of the total number of attribute values. In our experiments the attribute value of a tuple is taken as an integer. We refer to this integer value as the data value.

As an example, a typical run may consist of eight nodes, 8,000 data values, and a uniqueness of 0.4. The database randomly generates the required data and then distributes the data among the nodes, either uniformly (each node receives the same number of data values) or nonuniformly.

We used each of the three algorithms for parallel duplicate removal and recorded the execution times. The execution times may include the local duplicate removal time and the collection time, as described earlier. Since the data are generated randomly, we expect the execution time of an algorithm to change every time the experiment is carried out. To account for the random behavior, we conducted the experiments multiple times and observed the deviation of the execution times of an algorithm.

We found a small deviation in the execution times for all the runs except those with nonuniformly distributed data among the nodes. Therefore, for the data distributed uniformly among the nodes, we took the average of five runs as the true execution time of an algorithm. For the nonuniformly distributed data, we took the average of 25 runs of each algorithm as the execution time.

Our objective was to evaluate the performance of each algorithm under different work load and system conditions with the eventual goal of dynamically selecting the algorithm of choice for each application. The independent variables were

- the number of nodes,
- the number of data elements,
- the size of each element (or tuple),
- the uniqueness of the data (degree of skew),
- the initial distribution of the data, and
- the desired final distribution of the data (distributed or centralized).

Note that the choice of the optimum algorithm must be determined not only by the performance evaluation but also by the constraints on the initial data and final results.

Figure 9a shows the execution times (including the local duplicate removal times) of the three algorithms and an optimal algorithm as a function of the uniqueness. The system consisted of 16 nodes and 16,000 data values. The optimal

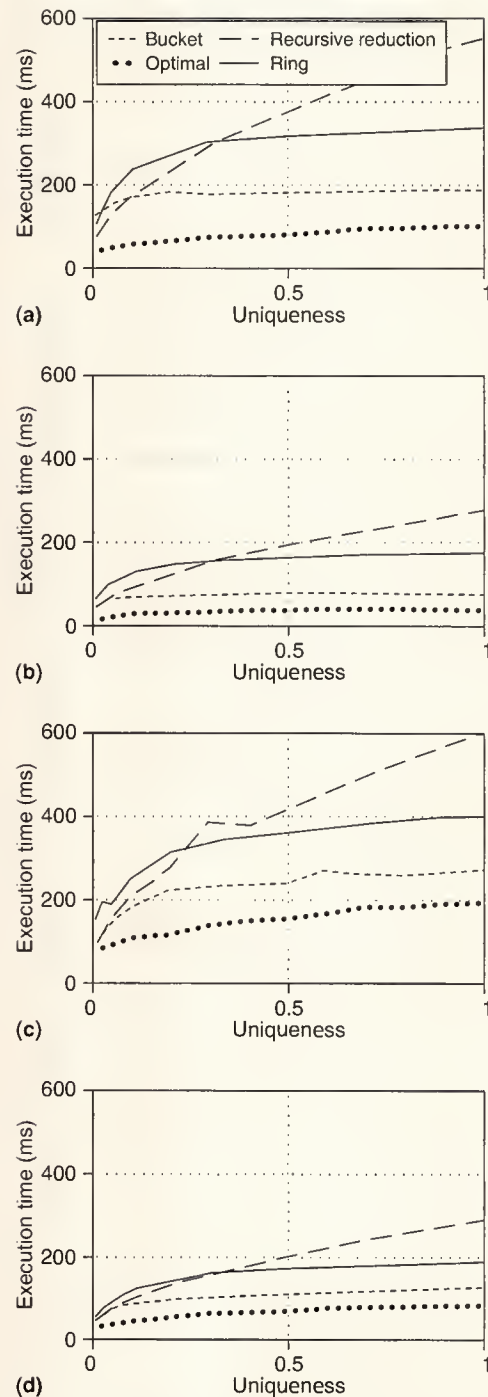


Figure 9. Execution times including the local duplicate removal times: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

algorithm using N nodes requires $(1/N)$ th the time taken by an optimal uniprocessor algorithm (a tree duplicate removal algorithm, in our case).

Note that for the bucket algorithm the speedup in the execution time is about $N/2$ when N nodes are used. Moreover, the speedup is the same for all values of uniqueness. The speedups for the ring and recursive reduction algorithms are small for high uniqueness values. But the speedups increase as the uniqueness decreases.

Figures 9b, 9c, and 9d show similar plots for 16 nodes, 8,000 data values; 8 nodes, 16,000 data values; and 8 nodes, 8,000 data values. The bucket algorithm is insensitive to the uniqueness of the data, except for very low uniqueness values. The recursive reduction is the most sensitive, and the ring algorithm is moderately sensitive to the uniqueness.

The bucket algorithm performs fastest for all except the very low values of uniqueness. For completely unique data (a uniqueness factor of 100 percent), the recursive reduction algorithm takes almost twice as long as the ring algorithm. But the performance of the recursive reduction significantly improves as the uniqueness decreases, due to its sensitivity to the uniqueness of data. At lower levels of uniqueness, the recursive reduction actually performs better than the ring. This threshold value of the uniqueness changes with the system and data conditions, namely the number of nodes and the number of data values. For example, for 16 nodes and 16,000 data values, the threshold uniqueness is 0.28, whereas for 16 nodes and 8,000 data values, it is 0.31.

Three components of the execution time explain sensitivity towards uniqueness.

- **Local duplicate removal.** Consider a database of size M with a global uniqueness p . Divide the database into N parts and let the local uniqueness of each part be q . We see that, in general, q will be larger than p . In fact, for $p \gg 1/N$, $q \approx 1$. For example, for $M = 16,000$ and $N = 16$, if p is reduced from 1.0 to 0.1, q remains at 1.0. Consequently, the local duplicate removal time is not sensitive to the global uniqueness p , especially for large values of p .⁷
- **Global duplicate removal.** This component consists of two parts:
 - Data transfer.* In the bucket and ring algorithms, the local uniqueness factors determine the size of packets to be transferred. For both algorithms, the packet size is not affected by change in the global uniqueness p . On the other hand, in recursive reduction, the size of the cube decreases as the recursion progresses and the local uniqueness changes from 1.0 (in the first recursion) to p (in the last recursion step). As a result, the size of the packets to be transferred increases from the first to the last recursion and is sensitive to the global uniqueness.
 - Local processing.* For the bucket algorithm, the size of

the received packet and the size of the local database are unaffected by the change in the global uniqueness. As a result, the local processing time of the bucket algorithm is insensitive to global uniqueness. For the ring algorithm, the size of the received packets and the initial size of the local database do not depend upon the global uniqueness. As the iterations progress however, the local database shrinks. The rate of this decrease depends on the global uniqueness. The lower the global uniqueness, the faster the local database shrinks. Hence, the local processing time is moderately sensitive to the change in the global uniqueness. For the recursive reduction algorithm, as explained earlier, both the size of the received packets and the size of the local database are sensitive to the global uniqueness. Consequently, the local processing time of the recursive reduction is highly sensitive to the global uniqueness.

Depending on the particular application at hand, the local duplicate removal time may or may not be important. If this time is not taken into account, the execution times of the three algorithms decrease by a fixed amount. Since the local duplicate removal time changes with uniqueness, the decrease in the execution time—though exactly the same for all three algorithms—differs for each uniqueness. Even if we do not consider the local duplicate removal time, the threshold value of the uniqueness, below which the recursive reduction algorithm performs better than the ring algorithm, does not change.⁷

The performance of the three algorithms for very low uniqueness values, shown in Figure 9, is of interest. Figure 10 shows execution times of the three algorithms for uniqueness varying from 0.001 to 0.01. Note that the ring algorithm is the most time-intensive of the three. The recursive reduction performs better than the bucket for most uniqueness values.

In the ring and the bucket algorithms, the final results are distributed among the different nodes. Such a distribution may be desired in various situations including intermediate relations within the execution of a query. In many applications however, the final results are needed in one node. In such cases, for the ring and bucket algorithms the distributed results must be collected and merged at one node. Note that the recursive reduction algorithm already includes the collection time.

Figure 11 shows the execution times including collection. The ring algorithm performs slowest. The bucket algorithm's performance degrades but remains better than that of the recursive reduction algorithm, especially for high uniqueness factors. For uniqueness values below a threshold, recursive reduction is better than bucket. The value of this uniqueness threshold depends on the number of nodes and the data size.

We have assumed that the data with particular uniqueness are generated randomly and that the distribution of different

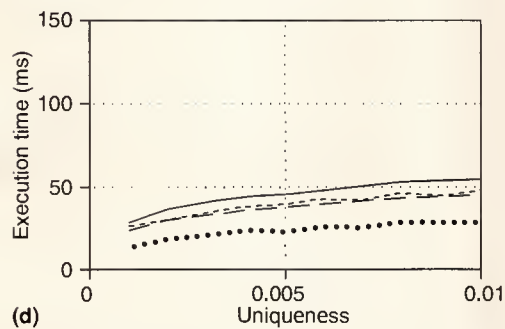
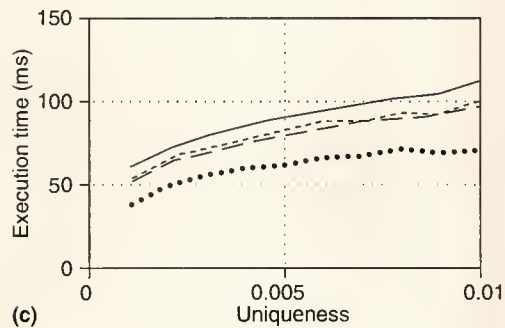
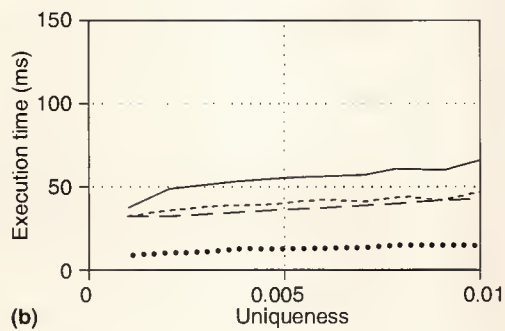
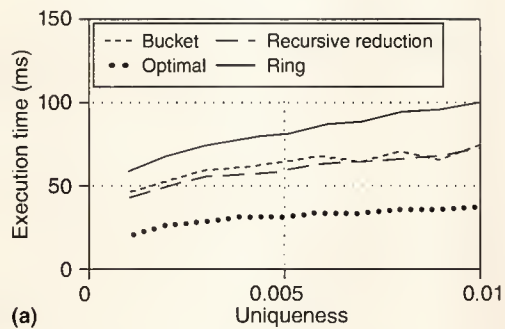


Figure 10. Execution times for low uniqueness, including the local duplicate removal times: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

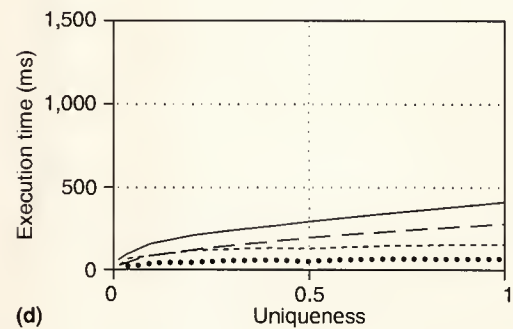
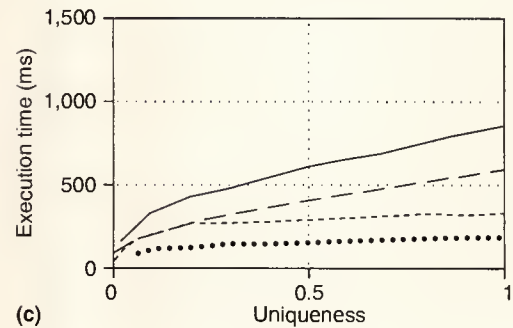
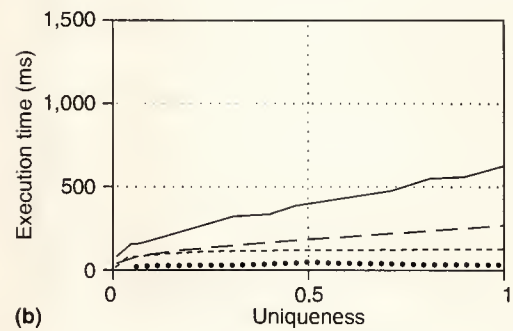
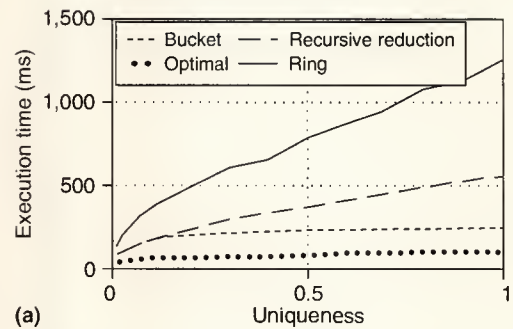


Figure 11. Execution times including collection: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

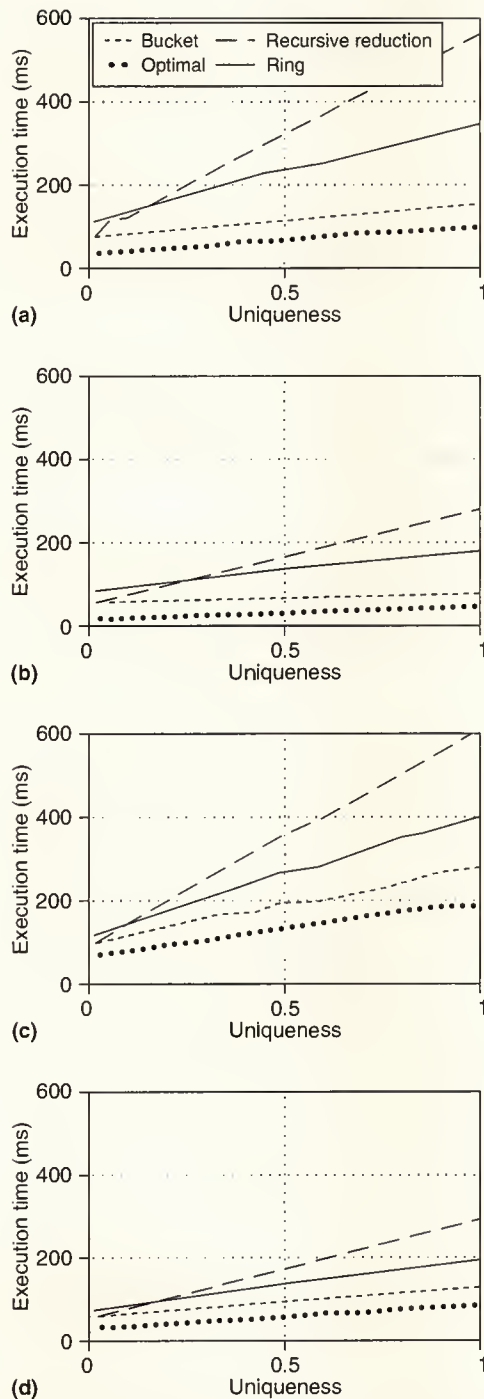


Figure 12. Execution times for nonuniform tuple counts: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

unique values in the data is uniform. This assumption may not always be true. We studied the effect of a skewed distribution of the unique data values on the algorithms by generating the data values using a nonuniform distribution. Figure 12 shows the execution times of the three algorithms when we generated data using a Gaussian distribution with μ equaling the mean of the unique values to be generated and σ equaling 50.

As expected, the execution times of all three algorithms remain unchanged for high uniqueness. For low uniqueness, the execution times of the ring and the recursive reduction algorithms decrease. This is not surprising because, if the distribution of the unique values is skewed, the local uniqueness factor on each of the nodes is generally smaller than the ones if the unique values were not skewed. Smaller local uniqueness implies that the local as well as the global duplicate removal times are smaller.

The performance of the ring and recursive reduction algorithms for the uniformly generated data are the worst case performance for a specified uniqueness as far as the skewness of the unique values is concerned. In the case of the bucket algorithm, the local duplicate removal time decreases, as it does for the ring and recursive reduction algorithms.

The global duplicate removal time increases because the design of the bucket algorithm assumes a uniform distribution of data. If the distribution of the data is skewed, the size of each bucket will differ. As a result, after a few iterations some nodes will be processing and transferring large amounts of data, whereas others may be operating on small amounts of data. The discrepancies among the volume of the data on different nodes increase with each iteration. Hence, in the bucket algorithm, the load-sharing among the nodes is not uniform if the data are not generated uniformly. This increases the global duplicate removal time of the bucket algorithm.

The total execution time will either decrease or increase depending on the sum total of the changes in the local and global duplicate removal times. Comparing Figures 9 and 12, we see that the total execution time remains almost constant for the bucket algorithm.

We assumed the distribution of the initial data among the nodes was uniform. This, too, may not be always the case. To study the effect of nonuniformly distributed data on the three algorithms, we uniformly generated data but distributed it nonuniformly to the nodes as follows:

- Generate a random number between 0 and N using a uniform random number generator.
- Let the number be x_0 .
- Randomly select x_0 data values and assign them to node 0.
- Now generate a random number between 0 and $(N-x_0)$ using a uniform random number generator.
- Let that number be x_1 .
- Randomly select x_1 data values and assign them to node 1.

- Repeat this procedure to assign the data to the remaining nodes.
- Assign to the last node all data not assigned to any other node.

As can be expected, the execution times of the three algorithms change substantially every time new data are generated using this procedure. Therefore, we repeated the procedure 25 times instead of the normal five times. Figure 13 shows the results on the nonuniformly distributed data. Note that the execution of all three algorithms increases by a factor of 2 to 3. However, the bucket algorithm remains the algorithm of choice.

Another important issue related to the initial data is the tuple size. So far we have assumed that the data consist of only the key, which is assumed to be an integer value. In reality, a tuple consists of a key along with other fields of varying sizes. In such cases, the duplicates are removed by appending the other fields of a duplicated tuple. For example, assume that a tuple consists of a key and a 10-byte field. If the data contain 50 duplicates of a key value, after the duplicates are removed, the key value will occur only once along with 50 fields of 10 bytes each.

Figure 14 on page 54 shows the execution times of the three algorithms for varying tuple sizes. Note that the execution times of all three algorithms increase by an order of magnitude. The recursive reduction is the most sensitive to the tuple size and clearly performs slowest for large tuple sizes. The sensitivity of the recursive reduction algorithm to the tuple size is consistent with its sensitivity to the uniqueness of p and is explained by examining the three components of the execution times, as discussed earlier. Table 2 on page 55 summarizes our results.

THE EFFICIENT EXPLOITATION OF PARALLELISM in a distributed memory architecture necessitates the development of algorithms designed for the underlying execution environment. Proper algorithmic design accounts for both the computational and communicational demands of the application.

Our study focused on algorithms designed for duplicate removal on a hypercube database system. We proposed, implemented, and evaluated three algorithms. We then used the results obtained by a performance evaluation of these algorithms under varying experimental parameters to determine the optimality of a given algorithm for a particular task.

Modeling the three algorithms and predicting their performance for a given system and load characteristics simplifies the determination of the algorithm of choice.⁷ Another advantage of using the analytical models is that the results can be generalized to different-size iPSC/2s, as well as to other types of hypercubes provided certain system parameters—namely, the processing and the communication speeds—are known a priori.

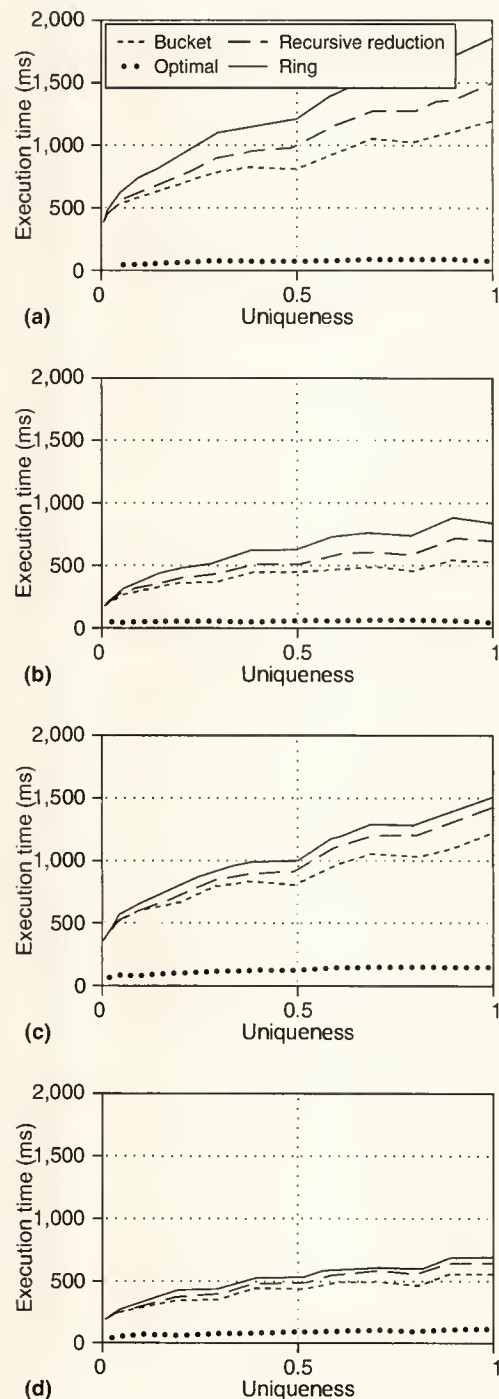


Figure 13. Execution times for data distributed nonuniformly among the nodes: 16 nodes, 16,000 data values (a); 16 nodes, 8,000 data values (b); 8 nodes, 16,000 data values (c); and 8 nodes, 8,000 data values (d).

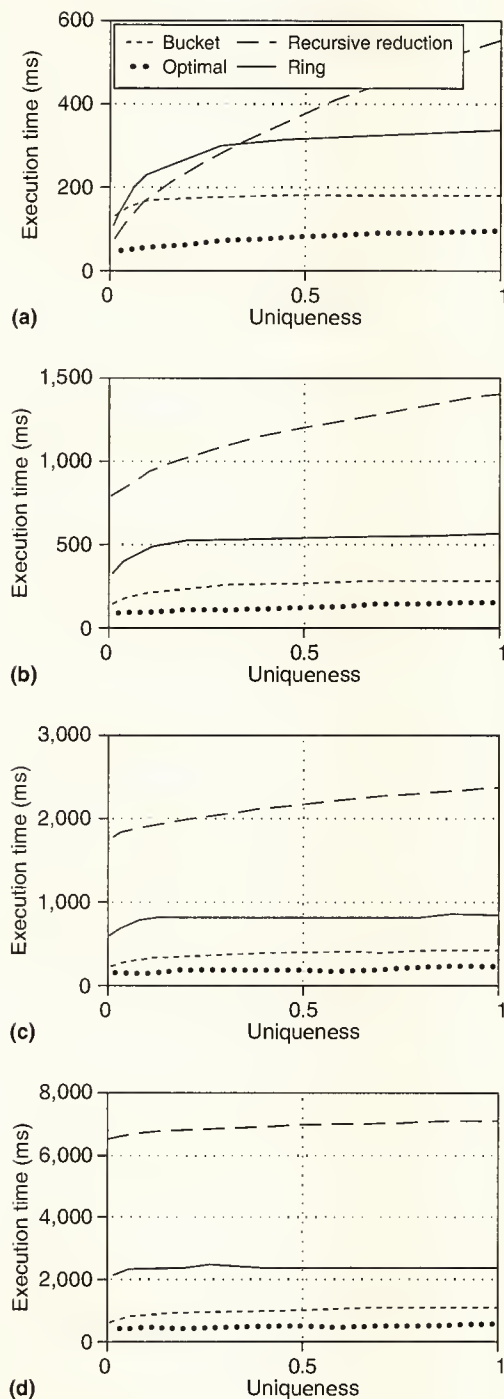


Figure 14. Execution times for different tuple sizes (16 nodes, 16,000 data values): 4-byte tuples (a); 14-byte tuples (b); 24-byte tuples (c); 104-byte tuples (d).

The distributed database is a special case of a multicomputer in which the communication bandwidth is lower than a tightly coupled multicomputer. Therefore, the analytical models of the three duplicate removal algorithms can be generalized to distributed database design as well.

We plan to design a dynamic query optimization scheme that incorporates the results of our study in determining how to best process a given relational operator within a given query. By examining the degree of uniqueness within the input relations and selecting the corresponding optimal duplicate removal algorithm, we intend to reduce the total query processing time. The cost of determining the optimal algorithm is an important consideration that needs to be addressed. ■

Acknowledgments

Grants from the National Science Foundation (a CISE Research Instrumentation Grant and a Research Initiation Grant), the Virginia Center for Innovative Technology, and Database & Communication Systems, Inc. supported this work.

References

1. C.K. Baru and O. Frieder, "Database Operations in a Cube-Connected Multicomputer System," *IEEE Trans. Computers*, Vol. 38, No. 6, June 1989, pp. 920-927.
2. D. Schneider and D. DeWitt, "A Performance Evaluation of Four Parallel Algorithms in a Shared-Nothing Multiprocessor Environment," *Proc. ACM/SIGMOD Conf., Assn. for Computing Machinery*, N.Y., 1989, pp. 110-121.
3. W.C. Wong, T. Suda, and L. Bic, "Performance Analysis of a Message-Oriented Knowledge Base," *IEEE Trans. Computers*, Vol. 39, No. 7, July 1991, pp. 951-957.
4. B. Arlid, W. Baugsto, and J.F. Greipsland, "Parallel Sorting Methods for Large Data Volumes on a Hypercube Database Computer," *Proc. Sixth Int'l Workshop on Database Machines*, Springer-Verlag, New York, 1989.
5. G. Fox et al., *Solving Problems on Concurrent Processors*, Prentice-Hall, Englewood Cliffs, N.J., 1988, pp. 327-348.
6. O. Frieder, "Multiprocessor Algorithms for Relational-Database Operators on Hypercube Systems," *Computer*, Vol. 23, No. 11, Nov. 1990, pp. 13-28.
7. V.A. Topkar, O. Frieder, and A.K. Sood, "Duplicate Removal on Hypercube Engines: An Experimental Analysis," *Parallel Computing*, Vol. 17, 1991, North-Holland, New York, pp. 845-871.
8. W.C. Athas and C.L. Seitz, "Multicomputers: Message-Passing Concurrent Computers," *Computer*, Vol. 21, No. 8, Aug. 1988, pp. 9-24.
9. J.P. Hayes et al., "A Microprocessor-Based Hypercube Supercomputer," *IEEE Micro*, Vol. 6, No. 5, Oct. 1986, pp. 6-17.

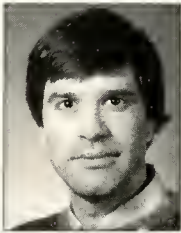
Table 2. Summary of the algorithm of choice.

Constraints		Algorithm of choice			
		N=16 M=16,000	N=16 M=8,000	N=8 M=16,000	N=8 M=8,000
Initial data not unique and final results may be distributed	Range of data Values known	Bucket for $p > 0.09$ RR* for $p < 0.09$	Bucket for $p > 0.04$ RR for $p < 0.04$	Bucket for $p > 0.02$ RR for $p < 0.02$	Bucket for $p > 0.02$ RR for $p < 0.02$
	Range of data Values not known	Ring for $p > 0.34$ RR for $p < 0.34$	Ring for $p > 0.32$ RR for $p < 0.32$	Ring for $p > 0.25$ RR for $p < 0.25$	Ring for $p > 0.3$ RR for $p < 0.3$
Initial data unique and sorted, and final results may be distributed	Range of data Values known	Bucket for $p > 0.1$ RR for $p < 0.1$	Bucket for $p > 0.04$ RR for $p < 0.04$	Bucket for $p > 0.02$ RR for $p < 0.02$	Bucket for $p > 0.02$ RR for $p < 0.02$
	Range of data Values not known	Ring for $p > 0.4$ RR for $p < 0.4$	Ring for $p > 0.3$ RR for $p < 0.3$	Ring for $p > 0.25$ RR for $p < 0.25$	Ring for $p > 0.3$ RR for $p < 0.3$
Initial data not unique and final results should not be distributed	Range of data Values known	Bucket for $p > 0.1$ RR for $p < 0.1$	Bucket for $p > 0.12$ RR for $p < 0.12$	Bucket for $p > 0.13$ RR for $p < 0.13$	Bucket for $p > 0.13$ RR for $p < 0.13$
	Range of data Values not known	RR	RR	RR	RR
Initial data not unique and final results may be distributed. Tuple counts not uniform	Range of data Values known	Bucket	Bucket	Bucket	Bucket
	Range of data Values not known	Ring for $p > 0.15$ RR for $p < 0.15$	Ring for $p > 0.25$ RR for $p < 0.25$	Ring for $p > 0.08$ RR for $p < 0.08$	Ring for $p > 0.15$ RR for $p < 0.15$
Initial data not unique and final results may be distributed. Data not distributed uniformly on the nodes	Range of data Values known	Bucket	Bucket	Bucket	Bucket
	Range of data Values not known	RR	RR	RR	RR
Initial data not unique and final results may be distributed. Tuple size=29 bytes	Range of data Values known	Bucket	Bucket	Bucket	Bucket
	Range of data Values not known	Ring	Ring	Ring	Ring

RR* Recursive reduction

p Global uniqueness

10. D. Bergmark et al., "On the Performance of the FPS T-series Hypercube," *Proc. Second Conf. Hypercube Multiprocessors*, in *Hypercube Multiprocessors*, SIAM, Philadelphia, 1987.
11. P. Close, "The iPSC/2 Node Architecture," *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, Jan. 1988, SIAM.
12. *iPSC/2 and iPSC/860 Users' Guide*, Report 311532-006, Intel Corp., Beaverton, Ore., June 1990.
13. S.F. Nugent, "The iPSC/2 Direct-Connect Communication Technology," *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, SIAM, Jan. 1988.
14. D. Maier, *The Theory of Relational Databases*, Computer Science Press, Rockville, Md., 1983.
15. L. Allen and O. Frieder, "Exploiting Database Technology in Medical Arenas: A Critical Assessment of PACS," to be published in *IEEE Engineering in Medicine and Biology* (Special Issue on Computers in Medicine), Mar. 1992.
16. O. Frieder, "A Parallel Database-Driven Protocol Verification System Prototype," to be published in *Software Practice & Experience*, Wiley Press, New York, 1992.
17. M. Abduelguerfi and A. K. Sood, "Computational Complexity of Sorting and Joining Relations with Duplicates," to appear in *IEEE Transactions on Data and Knowledge Engineering*. Also available in *Proc. Parbase 1990*, IEEE Computer Society Press, Los Alamitos, Calif., 1990.
18. M.S. Lakshmi and P.S. Yu, "Limiting Factors of Join Performance on Parallel Processors," *Proc. IEEE Fifth Int'l Conf. Data Engineering*, CS Press, Feb. 1989.
19. E. Omiecinski and E. Tien, "A Hash-Based Join Algorithm for a Cube-Connected Parallel Computer," *Information Processing Letters*, Vol. 30, No. 5, 1989, pp. 269-275.



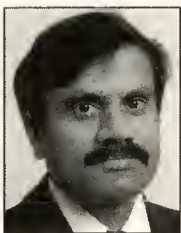
Ophir Frieder teaches computer science at George Mason University. His research interests include parallel and distributed architectures, database systems, operating systems, and medical imaging architectures.

Frieder earned a BS in computer and communications science, and MS and PhD degrees in computer science and engineering, all from the University of Michigan. He is a member of Phi Beta Kappa and IEEE Computer Society.



Vijaykumar A. Topkar is a PhD student in information technology at George Mason University. His research interests include signal processing, computer vision, parallel processing, simulation, modeling, and optimization.

He received a bachelor's degree from the Indian Institute of Technology in Bombay and a master's degree from the Indian Institute of Technology in Kanpur, both in electrical engineering.



Ramesh K. Karne is a PhD candidate at George Mason University. He works as an advisory engineer and scientist at IBM. His research interests are parallel simulation, massively parallel architectures, artificial intelligence and simulation, and intelligent hardware design.

Karne holds a BE in electronics and communication engineering from Osmania University in Hyderabad, India, and an MS in electrical and computer engineering from the University of Wisconsin at Madison. He belongs to the IEEE and the ACM.



Arun K. Sood is a professor of computer science at George Mason University. His research interests include image analysis, signal processing, parallel and distributed processing, database machines, performance modeling, and optimization.

Sood received a bachelor's degree from the Indian Institute of Technology in Delhi and MS and PhD degrees from Carnegie Mellon University, all in electrical engineering. He is a member of IEEE Systems, Man, and Cybernetics Society's Administrative Committee.

Address questions concerning this article to Ophir Frieder, Computer Science Department, George Mason University, 4400 University Drive, Fairfax, VA 22030; or via e-mail at ophir@gmuvmx2.gmu.edu.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 159	Medium 160	High 161
---------	------------	----------



Conformance Testing of VMEbus and Multibus II Products

Designers working with the European Community's Conformance Testing Services program have developed a system for testing VMEbus and Multibus II products. Conformance is necessary to allow and guarantee interoperability between products from different vendors. The EC's test system is mainly automated to reduce costs and ensure impartiality. An overview of the system's components familiarizes readers with its procedures.

Marcus Adams

Yi Qian

Jacek Tomaszunas

*Computer Research Center
(FZI), Karlsruhe*

Josef Burtscheidt

Edgar Kaiser

*Central Laboratory for
Electronics, Jülich*

Csaba Juhasz

*Technical University of
Budapest*

Conformance testing seeks to determine if a product or parts of a product (in our case, parallel bus interfaces) has been built according to the rules of the applied standard.¹ One way to check conformity is to examine the schematics, programmable logic array data, and PROM data, and compare the results with the rules of the standard. Another way is to create a testing environment representing the standard and test the product against it. Dedicated devices of the test environment monitor the results of interaction.

This second way requires a big investment in methodology, hardware, and software before the first product can be tested. But this approach's advantage is that it produces test results that are objective, reproducible, and independent from the test engineer. A consortium of advisors chose the second method for the Bus Interface Conformance Test (BICT) project.

Test system

The BICT system tests the conformity to standards of bus interfaces of VMEbus and Multibus II boards and their backplanes.^{2,3} The test system represents the standards and offers all the options the standards give. This approach allows a board to be plugged into the test system where predefined automatic or semiautomatic procedures test the features of the unit under test.⁴

The bus standards specify four main categories:

- timing and logical behavior of signals,
- message-passing protocol (for Multibus II only),
- electrical properties of boards and backplanes, and
- mechanical dimensions of boards and backplanes.

A specific configuration of the test system and its devices is required for each category. Figure 1 on page 58 shows the general configuration of the BICT system, including the dial bench gauge, which is operated manually.^{5,6}

A test system controller manages the activities of the test system devices and gives instructions to the operator, or test officer. For each test campaign the controller collects the measuring results, stores them in a database, and generates a test report on the basis of collected data. The controller performs a series of functions, (see box on page 58) classified by type of action.⁷

In this article we explain a test campaign in the sense of a walk through the test system, in the way a customer—such as a manufacturer, reseller, or original equipment manufacturer—would see it. We focus on testing bus interfaces because it is more complex than testing backplanes.

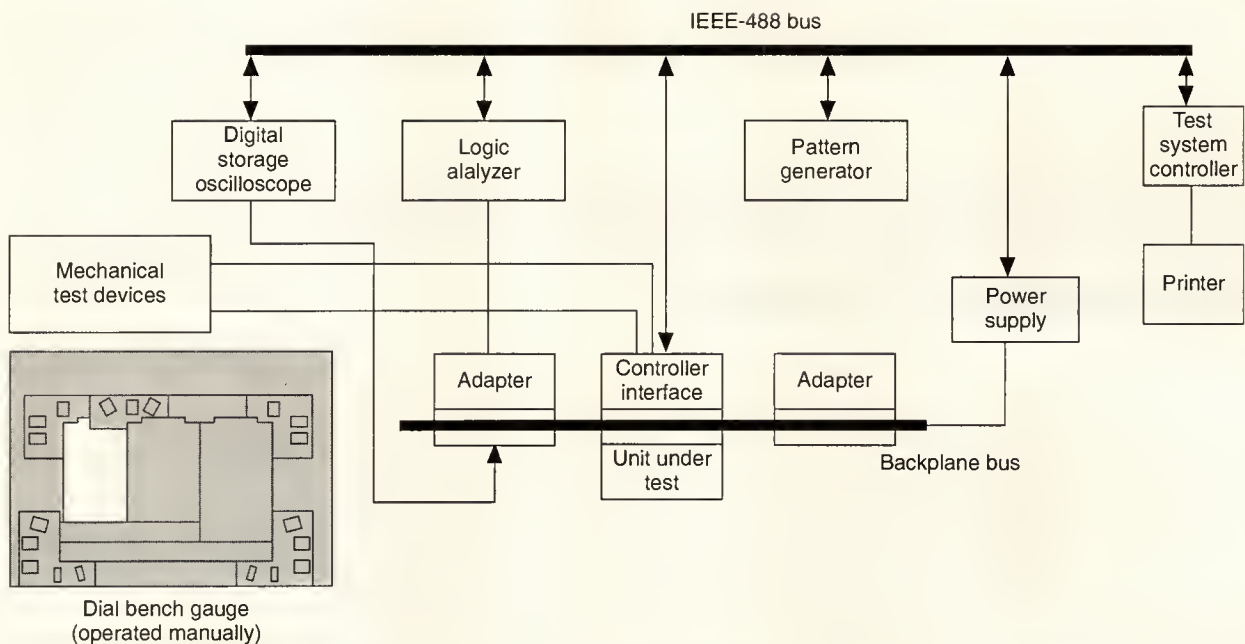


Figure 1. BICT system.

Test system controller functions

For the classifications listed here, the controller provides the following information or performs the following tasks. In the case of manual tests, the operator provides the information, which the controller then analyzes.

The test officer

- Information about the actual test and test state
- Instructions about mechanical tests to be carried through by the officer
- Information about test results by immediate check against parts of the relevant standard
- Requests test results and offers data entry capabilities

Test system controller

- Start/stop logic analyzer
- Start/stop pattern generator
- Poll logic analyzer and digital storage oscilloscope on trigger or time-out if no triggering
- Select and start test routine of the unit's upper tester (via communication channel)
- Upload data from test devices
- Download data to test devices

Digital storage oscilloscope

- Measurement of signal edges and waveforms for electrical test
- Measurement of voltage levels for static electrical test

Logic analyzer

- Run the trigger to monitor the logical and timing behavior of the unit
- Store captured data in case of triggering
- Access to database of trigger setups
- Trigger on certain messages or bus states
- Store captured message in case of triggering

Pattern generator

- Run sequence of patterns for bus operation initiation/response
- Control of delay lines for signal adjustment

Power supply

- Provide unit with current in the specified voltage range
- Measurement of current consumed by unit

Mechanical test devices (dial bench gauge, slide calipers)

- Measurement of boards
- Measurement of devices mounted on boards

BICS/BIXIT

Before testing begins, a customer provides the following information about the unit on two DOS-based, electronic forms:

- **Bus interface conformance statement (BICS).** The customer enters the capabilities and options of the standard implemented in the unit.
- **Bus interface extra information for testing (BIXIT).** The customer provides further details on the implementation features of the unit, such as address ranges, interrupt levels, bus request levels, and message types.

BICS/BIXIT is a menu-driven, interactive input program that automatically checks for completeness and consistency as the user fills in the form. It features dynamic data management and automatic data interfacing and recording. Figure 2 depicts a BICS/BIXIT input screen requesting information about the implemented features of a memory slave.

Chip-level testing is complex, time-consuming, and not cost-effective for the purpose of conformance testing. Board manufacturers use relatively few standardized silicon devices to implement the interface drivers, and sufficient and reliable information about their capabilities is available. We can therefore avoid testing on the chip level.⁸

In lieu of such testing, the board manufacturer provides information about drivers, transceivers, and other devices that implement the bus interface, on another DOS-based form, the vendor claim. The vendor returns this form, along with the BICS and BIXIT data, to the center, where it is analyzed by a test program running on the controller.

In the first step of testing, static conformance review, the controller checks this information against a silicon database and the relevant rules of the standard.

Mechanical test

Test officers manually assist in the next step, testing whether the board fits into a standard rack and its backplane. A mechanical test program running on the controller guides the tester and indicates the points to be measured. The mechanical test is based on two test methods: the outline test with the dial bench gauge and the rack fit test with the rack simulation.

For the outline test, the dial bench gauge (see Figure 1) can be configured according to the controller's instructions and the test officer mounts the unit board onto the gauge. The tester then applies the test tool to the measuring points asked for by the test program. The read values

transmit via the IEEE 488 bus to the controller. The program determines if the read value is plausible and compares it with the requirements of the standard.

The dial bench gauge consists of a base plate on which smaller plates are mounted, carrying a reference bolt for each position to be measured. The positions of these reference bolts vary for different board sizes (single, double, and triple European cards, 160 or 220 mm long).

The tester fixes the unit on two clamping strips. Before taking the measurements, the tester samples a reference board to get the offset values. This procedure, called the calibration phase, delivers the offset factors, which are stored in the controller and integrated into the measurement calculation. Figure 3 on page 60 shows the configuration for the mechanical test.

Testers use a slide caliper for special cases where the depth of the gauge is not applicable. The rack fit test is a purely qualitative test, as postulated in the VME standards. The tester performs this test manually.

Electrical test

The electrical test verifies rules concerning requirements for electrical characteristics and behavior. The measurements are electrical, chronological, or a mixture of the two.

One part of the electrical test measures electrical characteristics independent of special capabilities of the unit, such as connectivity, termination networks, and power consumption. The other part times signals or performs specific functions of the unit, measuring signal waveforms (rise and fall times, high and low times), signal levels, and power on/off behavior.

BICS/BIXIT Edit	
File	Edit View Check Document Quit
Master #1	
Local Identifier : 68020 CPU	
Cycle Type	
Basic Data Transfer : Y	
Block-Transfer : N	
Read-Modify-Write : Y	
Unaligned Transfer : Y	
Address-Only : N	
Local Bus Timer (BIXIT) : not implemented	
Data Width	Address Width
D8 : Y	A16 : Y
D16 : Y	A24 : Y
D32 : Y	A32 : Y
BIXIT :	Remarks :

Figure 2. Input screen of BICS/BIXIT.

As with the mechanical test, the controller guides the operator through the electrical tests (see Figure 4). The test officers handle the test probes manually, since for each rule different test points must be checked. The obtained results are uploaded to the controller via an IEEE 488 interface, where a program compares them with the specification values and generates a statement for the report.

The electrical test does not measure driving and receiving characteristics of the signal lines. Testing them would mean testing silicon and, as mentioned earlier, this exceeds the scope of conformance testing on the board level. Therefore, the test officers refer to the information provided on the vendor claim form to check these devices or assemblies. Operators apply this method when items are not testable or may change from board to board. Items tested in this way include used drivers, receivers, and connectors.

Timing protocol test

The timing protocol test determines whether the interface modules of the unit communicate through the bus with other modules in accordance with the relevant timing protocol given by the test specification. (Interface modules include the interrupt handler, interrupt requester, and arbiter.) The system individually tests every module or joined modules of the unit.

A pattern generator stimulates the module(s) under test through the bus, while a logic analyzer detects anomalous (or, illegal) events on the bus (see Figure 5). The stimulating patterns created on the bus by the pattern generator, as well as the responses created on the bus by the module(s) under test, constitute a bus operation.

The standard rules define the modules and their options, or capabilities. Modules can participate in different bus cycles according to their defined capabilities. For example, a VME master module with 16-bit data transfer capabilities may participate in single- and double-byte read-and-write data transfers, as well as block transfers. But it may not participate in Byte (0-3) data transfers.

A module option may also define a cycle option. A VME arbiter module may resolve a request according to a standard-specific algorithm (single, priority, or round robin) and a VME requester may release the bus according to one of two algorithms (release-when-done or release-on-request).

Bus operation. Every bus operation is specified as a set of bus cycles and hence as a set of timing and data protocol test rules. Part of the bus operation stimulates the test module. The other part is the response of the unit, which will be analyzed by the logic analyzer.

For example, a CPU module reads data and writes it to a

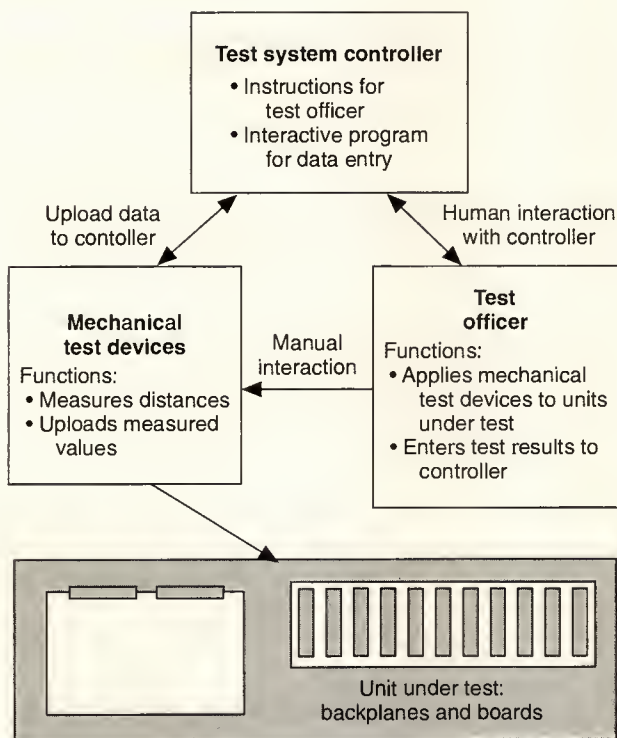


Figure 3. Mechanical test.

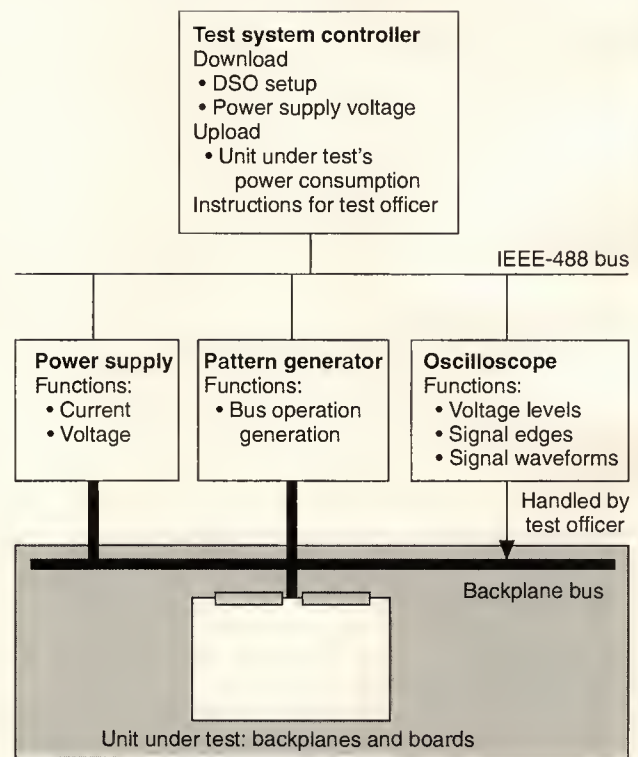


Figure 4. Electrical test.

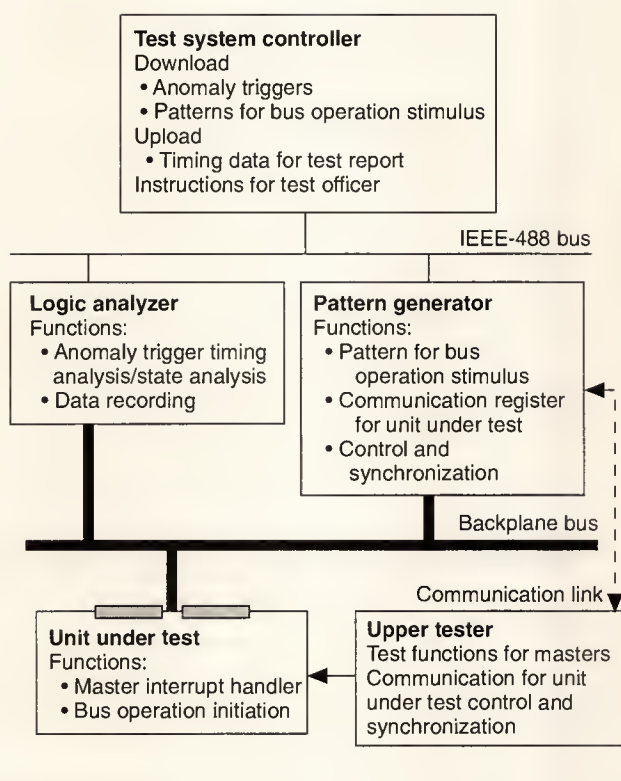


Figure 5. Timing protocol test: test of masters.

memory board. First it runs an arbitration cycle to gain the bus. Next it runs a read cycle to fetch the data from the module. To write its data to the module, it performs a write cycle. Finally, the CPU module releases the bus. In this example the bus operation consists of the arbitration, read, and write-and-release cycles. In every cycle the CPU module (stimulus) gives signals that are driven by the memory module (response).

Pattern generation. The signal relations, into which timing rules are translated, define the limits of the stimulating rules. For worst-case timing protocol testing, we developed worst-case stimulating patterns. For basic interconnection timing protocol testing, the system stimulates the unit using more tolerant limits.

A pattern generation program, reads signal relations associated with a bus cycle as input information and translates them to stimulating patterns and wait statements for the pattern generator. Data protocol test rules are taken into account by the pattern generation program, as needed.

Real-time analysis. The real-time analysis method tests behavioral characteristics (or, functional tests) and verifies the relevant rules. Real-time analysis facilitates exhaustive testing and supports the BICT philosophy:

Many complete bus cycles/operations perform consecutively. The logic analyzer monitors the signals on the bus in real time and only freezes data in its memory if a trigger condition is fulfilled. While cycles run, the controller downloads different trigger words to the logic analyzer.

We decided to run the bus cycles/operations several times because 1) asynchronous behavior leads to deviations in reaction times, and 2) different states of the unit may cause different reactions. This method uses anomaly triggers, so that only in cases where the standard is violated does the logic analyzer upload the data to the controller for the test report.

Control-and-status register test

The control-and-status register set holds system information in a standardized way. During a system startup, the configuration program uses it for systemwide initialization, configuration, and diagnostics.

The test for Multibus II checks the content, structure, and predefined functions of the control-and-status register. Since almost every item tested is a different case, we established an individual test procedure for each rule. During the test these procedures are executed one by one, according to the unit's capabilities.

Message-passing protocol test

The message-passing protocol describes a communication and data exchange procedure between agents using well-defined messages. We call it "network in a crate" because it applies mechanisms similar to those used in LANs. One message packet is transferred using one sequential data transfer bus operation in the message address space. In general, we define two types of messages.

- *Unsolicited.* An unsolicited message is a one-way transfer of a data packet to a destination address. The packet carries up to 28 bytes of data. Systems use unsolicited messages for interprocessor communication, very short data transfers, and virtual interrupts. Within one Multibus II system, we can define up to 255 message (interrupt) sources and 255 message (interrupt) destinations. The broadcast message, a special unsolicited message, is a simultaneous, one-way transfer of an unsolicited message to all available destination addresses in a system.
- *Solicited.* A solicited message transfer is a specified, finite exchange of data packets that transfers up to 16 Mbytes of data. Before such a message is sent, the system negotiates to ensure there is enough free data buffer available at the destination and to define other transfer parameters. The solicited message fragments into small

data packets, each containing at least 32 bytes. The system then sends them over the bus, one at a time.

The message-passing protocol test for Multibus II checks that the unit performs the sending and receiving of unsolicited and solicited messages correctly. For solicited messages, it also tests the negotiation of the transfer and the correct application of the negotiated parameters. In addition, the content of each message is checked.

Since the message-passing protocol uses a bus protocol on signal level, the unit must pass the electrical and functional test (on signal level) successfully before operators perform the message-passing protocol test.

To perform this test, the vendor installs an upper tester which substitutes for the software layer above the unit. Figure 6 shows that a test routine runs through a communication channel between the controller and the upper tester.

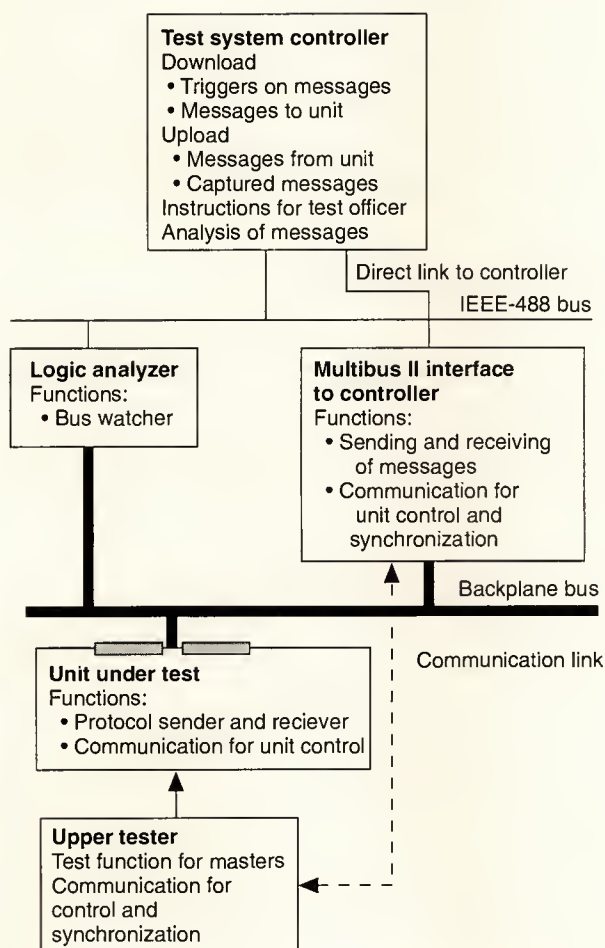


Figure 6. Message-passing protocol test.

BICT divides the message-passing test into two entities, each applying different analysis methods.

- *Messages sent.* The controller uses two methods to analyze outgoing messages. In most cases, the system uses a software analysis method. The Multibus II interface receives the unit's messages, which are then analyzed by the controller. But in some cases, in which the logic analyzer must monitor the bus, the system uses a bus analysis method. This method measures, for example, the duty cycle during a solicited message transfer, or detects unexpected messages sent by the unit.
- *Messages received.* Two methods can be used to test the message-receiving capability of a unit. In the on-board message verification method the upper tester analyzes the messages received by the unit and reports the test result to the controller. Since this will blow up the upper tester and with it the vendor's investment in installing it, we prefer the message reflection method. Here, the messages sent to the unit are reflected by the unit and analyzed by the test system.

To ensure that detected errors are caused by the unit's receiving capability and not by its sending capability, the unit must pass the sending capability before the test officer applies the message reflection method.

Test example

To demonstrate how the conformance test is carried out in practice, we describe here a message-passing protocol test case for Multibus II. Rule 13.5.1.2, Section 5 of the Multibus II test specification states, "If an agent is not able to receive a requested solicited message, it must send a buffer reject message to the agent that has sent the buffer request message." This rule checks the negotiation of a solicited-message transfer. The buffer reject message is an unsolicited message that sets up a solicited-message transfer. A system refuses the buffer request if, for example, a receiver does not have enough resources or the resources are, at that moment, not available.

Verification procedure. To test this rule, we apply the software analysis method. At the beginning of the verification procedure the controller, via the communication channel, starts a test routine of the upper tester installed on the unit. The selected test routine is responsible for receiving solicited messages.

Then the controller, via the Multibus II interface, sends a buffer request message to the unit (see Figure 7). In this case the requested length of the solicited message, 10 Mbytes, is greater than the available local resources of the unit, 1 Mbyte. The size of the unit's local resource has already been declared on the BICS/BIXIT form, so the controller can automatically calculate the requested length of the message.

Now the controller waits until the Multibus II interface re-

ceives a message. If no message arrives within a specific time, the test of the rule has failed. If a message arrives, the controller verifies whether the message is a buffer reject.

Results. At the end of the verification procedure the controller interprets the results and appends them to a log file. In the case of rule 13.5.1.2 section 5, it assigns a "failed" result if the unit sends no message or a message that is not a buffer reject.

Test report. After the test, the center prepares a report (see Figure 8) and sends it to the customer. The report describes the unit, the test suite, and all useful information concerning the executed tests and their results. The report's first section summarizes the number of tests applied and failed. A later section details the applied tests and the test sequence. If a rule was violated, the report notes the error and gives the corresponding rule text. A description of the test method applied can be found in the test specification. Customers can obtain more detailed data on the test execution and results, such as captured waveforms or data, from the test laboratory if needed for further analysis.

THE BICT SYSTEM OFFERS vendors the security of an impartial test of a board's conformity to bus standards. Conformity is a necessary precondition for the interoperability of boards sharing a bus.

To test a product, a center needs information only about the features and parameters of the board, not design information such as schematics, PROM, or PLA data. Design information stays with the originator.

The mainly automated test procedures ensure cost- and time-effectiveness and guarantee objective results. A basic aim of the BICT project was to eliminate as much human influence as possible. The results depend only on the strengths and weaknesses of the test system. We are optimistic that any remaining weaknesses will be eliminated as the test system matures.

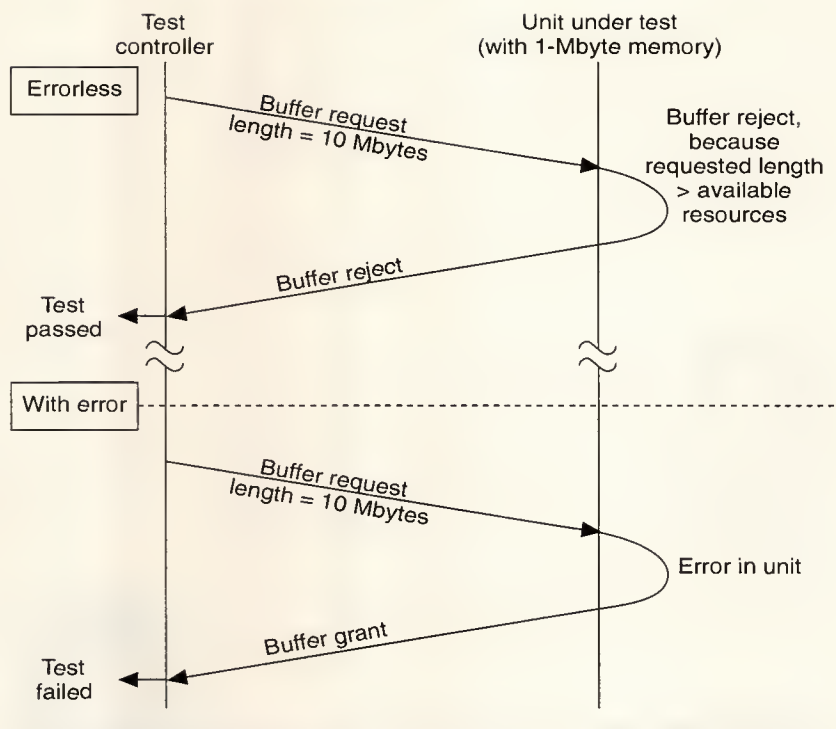


Figure 7. Test of negotiation phase (unit as receiver).

Message passing protocol test

The following errors were identified:

Rule number	Start		End		Result
	Date	Time	Date	Time	
13.4-1	910522	083910	910522	083910	passed
13.5.1.2-4	910522	083910	910522	083911	passed
13.5.1.2-5	910522	083956	910522	083957	failed
*** buffer grant message received from UUT					
13.5.1.2-7	910522	084002	910522	084003	passed

Details of failed tests

The following test cases failed:

Rule 13.5.1.2-5

If an agent is not able to receive a requested solicited message, it must send a buffer reject message to the agent that has sent the buffer request message.

Figure 8. BICT report section.

Starting this year, the service will be available at the test centers involved in the BICT project: the VDE Test and Certification Institute in Offenbach, Germany; S. Seferiades and Associates (SSA) in Athens; and Istituto Italiano del Marchio di Qualità (IMQ) in Milan. □

Acknowledgments

The European Community's Conformance Testing Services Program, Phase 2, (contract number 87/12227) sponsored this project.



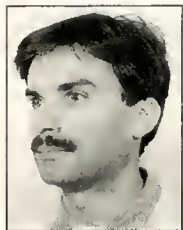
Marcus Adams heads the microcomputer technology department of the Computer Research Center (Forschungszentrum Informatik) in Karlsruhe. In 1986 he founded the German chapter of the VMEbus users association.

Adams received the Diploma in electrical engineering and a PhD in informatics from the University of Karlsruhe. He is president of the German VMEbus chapter and belongs to the German Electro-technicians Commission and the Association of German Electrotechnicians (VDE).



Yi Qian is a visiting scholar at FZI. His research interests include microprocessor systems, bus conformance test, pattern generation, and automatic test and measurement.

Yi holds a BS in electrical engineering and an MS in computer engineering from Peking University for Posts and Telecommunication, where he is a PhD candidate.



Jacek Tomaszunas is a visiting scholar in the department of microcomputer technology at FZI. He earned the Diploma and MS degrees in electrical engineering from the Technical University of Wroclaw in Poland.



Josef Burtscheidt works on the BICT project at the Central Laboratory for Electronics at KFA Research Center in Julich. He has developed and designed experiment control and microprocessor systems.

Burtscheidt holds a Diplom Ingenieur in communications and information processing from the Technical High School of Cologne.

References

1. *Information Processing Systems - OSI Conformance Testing Methodology and Framework*, ISO/IEC DP 9646-1,-5, Part 1, 1988-01-09 and Part 5, 1988-06-22.
2. *IEEE 1296 Standard for a High-Performance Synchronous 32-Bit Bus: Multibus II*, 1988-02-08.
3. *IEEE Standard for a Versatile Backplane Bus: VMEbus*, ANSI/IEEE Std 1014-1987, 1987-09-11.
4. BICT consortium, "Test Methodology," Tech. Report, European Commission, Brussels, Feb. 1990.
5. BICT consortium, "Test Tool Description," Tech. Report, European Commission, Brussels, Feb. 1990.
6. BICT consortium, "Final Test Tool Description," Tech. Report, European Commission, Brussels, Apr. 1991.
7. BICT consortium, "Test Bed Description," Tech. Report, European Commission, Brussels, Apr. 1991.
8. BICT consortium, "Board Test Procedures," Tech. Report, European Commission, Brussels, Oct. 1990.



Edgar Kaiser also works on the BICT project at the Central Laboratory for Electronics. He, too, has developed and designed experiment control and microprocessor systems.

Kaiser earned a Diplom Ingenieur in electronics and information processing from the Technical High School of Aachen.



Csaba Juhasz is a PhD candidate in process control at the Technical University of Budapest. He served as a research assistant on the BICT project at FZI.

Juhasz earned an MS in electrical engineering from TU, Budapest. He is currently a visiting scholar at the Department of Systems Science at The City University in London.

Address questions concerning this article to Marcus Adams, Forschungszentrum Informatik, Computer Research Center, Haid-und-Neu-Strasse 10-14, W-7500 Karlsruhe, Germany; or via e-mail at adams@fzi.uka.de.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 162

Medium 163

High 164

Micro Review

Richard Mateosian

2919 Forest Avenue

Berkeley, CA

94705-1310

(510) 540-7745

How I spent my Christmas vacation

My holiday tradition is to attack the big projects that I've been putting off all year. This year I plunged into the Macintosh.

I have a Mac SE/30. I use it every day in my consulting work. I also use it for reviewing the Macintosh software that I receive in abundance. This is the source of the problem.

In my consulting work I do simple word processing. I occasionally transmit text files over phone lines. My Mac SE/30's original memory size of 2 Mbytes was more than adequate for these tasks.

I enjoy reviewing new software, since the process forces me to learn to use powerful new tools that my natural inertia would otherwise keep me from exploring. Unfortunately, new software grows by leaps and bounds, and I can't keep up with it unless my computer does the same. Thus, my project for the holidays was to upgrade my Macintosh to 8 Mbytes.

Memory upgrade

Looked at objectively, a Macintosh memory upgrade is a pretty small project. It takes less than half an hour, but somehow I managed to put it off for nearly four months after I had everything I needed in hand. I probably would never have done it at all if it had not been for the idiot-proof package put together by the mail-order company called Mac Warehouse.

I called Mac Warehouse's toll-free number (800) 255-6227 to order the 1-Mbyte single in-line memory modules (SIMMs) required to upgrade a Macintosh. As I write this, their advertised price is \$59 each. I needed eight of them to upgrade from 2 Mbytes to 8 Mbytes. Each one replaces one of the original eight 256-kilobyte SIMMs. The Mac Warehouse people told me I

needed a simple toolkit, including a grounding wrist strap, which they sold me for \$9. They included a video cassette demonstration of the upgrade process. The entire package showed up at my house the next day. Then it sat in my office for more than three months.

The video is marvelous. I am about as unmechanical as a graduate of an engineering school could possibly be. Removing the SE/30's motherboard would have been beyond me with only written instructions. A woman with a reassuring manner demonstrated all of the simple but hard-to-explain-in-print steps required to upgrade any Mac's memory. With exquisitely manicured hands she deftly manipulated tools, eased wire harnesses out of hard-to-reach sockets, and snipped a resistor lead. She carried on a running description of what she was doing, calmly mentioning the inherent dangers at certain points. My favorite was when she said, "If you crack the video tube, it *will* implode."

I carried out the entire task in about a half hour. I sat on my living room rug with the grounding strap on my left wrist, the VCR remote control unit by my right hand, and the TV, my Macintosh, and my tools in front of me. The video runs 11 minutes, but I kept replaying key parts until I was sure I understood what I was doing. When I reassembled my Mac, everything worked perfectly.

Operating system upgrade

I would never change my operating system software if I didn't have to. Unfortunately, new software invariably makes old operating systems obsolete. When I loaded the latest version of Mathematica onto my Macintosh, it immediately announced that it wouldn't work on the version of the Macintosh operating system that I was

running. I could have put in a minor upgrade from 6.03 to 6.07, but I decided to switch to System 7, Apple's long-awaited major upgrade.

The Berkeley Macintosh Users' Group (BMUG) is a wonderful resource for anyone who uses Macintosh computers. When I decided to upgrade to System 7, I called them at (510) 549-BMUG. I bought their package deal, which includes the 10-disk upgrade set from Apple and *The Little System 7 Book* by Kay Yarborough Nelson (Peachpit Press, Berkeley, 1991, 158 pp., \$12.95), all for \$25. I did this about the same time I bought the SIMMs, and everything sat around my office for just about as long.

System 7 is quite different from previous Macintosh operating systems, but by the time I sat down to use it I felt right at home. I attribute this to the quality of the upgrade package that Apple put together, the excellence of *The Little System 7 Book*, and the many informative articles about System 7 in the *BMUG Newsletter*.

Users have come to expect installation programs to guide them through the installation of large application packages. While installation programs have certainly improved considerably over the last few years, Apple's upgrade software is a cut above any other that I've seen. It is a model for others to follow. Especially impressive were two Hypercard stacks designed to familiarize users with the new features of System 7. The one dealing with the new networking features is especially impressive. It simulates the behavior of the system, allowing you to make menu selections, click buttons, and so on, just as you would if you were performing the networking functions it is teaching you.

While my move to System 7 was easy, it also had its rough spots, all of which hinge on compatibility. Because the underlying software of System 7 differs substantially from previous versions, many software packages designed for previous versions will not

run. The Apple upgrade program begins by printing a list of all of the applications on your machine. It labels each one as "fully compatible," "mostly compatible," "must upgrade," or "information not available." Unfortunately, the last two categories predominated on my machine. The only ones listed as fully compatible were Word, Excel, Mathematica, and Mac Draw II.

***System 7 differs
from previous
Mac operating
systems; but I felt
right at home
with it.***

The compatibility printout contains helpful information on upgrading old software. It lists your version number and that of the first compatible version. It gives the phone number of the manufacturer of each package you need to upgrade. In some cases it indicates that the upgrade package actually includes a compatible version. Unfortunately, it appears to say that the upgrade package includes a compatible version of Hypercard, but this is only true if you buy the System 7 Personal Upgrade Kit from Apple. My BMUG package did not contain Hypercard, and the Hypercard version that came originally with my SE/30 is incompatible with System 7. So I had to scurry around to obtain a working version.

While simple incompatibility with the programming of System 7 is a problem for most packages, some have an even worse problem. System 7 makes a large part of what they do obsolete. The popular Suitcase II and MasterJugger programs are prime examples. Many of their capabilities for managing fonts, sounds, and desk accesso-

ries are no longer needed. System 7 handles sounds and fonts by allowing you to drag their icons into or out of the system file. It eliminates the distinction between applications and desk accessories altogether, since it now incorporates the features of Multifinder. The apple menu simply lists the contents of a folder in the system file. The apple menu folder can contain any application, document, or folder. Selecting an item from the apple menu causes the item to be opened.

Another important feature of System 7 makes front-end programs like Power Station less important. System 7 allows you to create an alias for any document, application, or folder. The alias consumes a tiny amount of disk space and serves as a pointer to the actual item. Thus, for example, you can leave Excel or Word in its own folder but place an alias for each in the apple menu folder or the start-up folder. You can make aliases for the apple menu folder and the system file and leave the aliases on the desktop. I made an alias for the text file that contains this column and placed it in the apple menu folder. When I select "Column" from the apple menu, the system opens this text file and the corresponding word processor.

One important Power Station function that I haven't figured out how to do in System 7 is to associate an icon with an application and a specific set of documents to be opened with it. I would certainly keep Power Station under System 7 for its document-handling capabilities alone, but unfortunately my version caused the system to crash when I tried to run it. The Apple compatibility checker provided no information about Power Station, so this will take additional work on my part. I encountered a number of other small nuisances like that during the upgrade, but on the whole I'm quite happy.

As I write this column using Microsoft Word running under System 7, I am simultaneously also running Mathe-

matica, Hypercard, and the Variable Symbols Mathematica Help Stack (see Micro Review, Aug. 1991). I even have another 250 Kbytes left over in my new 8-Mbyte memory. I'm impressed. I wonder how long this euphoria will last.

Books

The holidays are also a nice time to sit by the fire reading, so I managed to look at a number of good books. One of the best of these (see next paragraph) is a little old for a computer book, and it does show its age in places, but it is so good that it's worth reading anyway. I hope the publishers bring out an updated version.

The Sachertorte Algorithm and Other Antidotes to Computer Anxiety, John Shore (Viking, New York, 1985, 286 pp., \$16.95)

Shore states his purpose to be promotion of a general understanding of computers. As a reflective and witty old hacker, he is well qualified to achieve that purpose. It looks to me as though he has done so, although I can't look at the book through the eyes of the intended audience. All I can say is that again and again he makes points that I'd want to make to an interested and intelligent beginner. It's hard to communicate the overall effect of this by quoting isolated phrases, but here are a few that appealed to me:

In general, jargon-filled error messages are symptoms of a basic problem, namely that the designers and programmers of many office and personal computer systems have failed to separate their own concerns from those of the user.

If you buy a new car and spend the next year having the dealer fix things that didn't work right to begin with, you don't say that your car is being maintained; you say you

bought a lemon. By this criterion, most software products are lemons.

While standing recently in a cold shower, I had occasion to think about replacing my hot water heater. Because I had been thinking about software engineering before the water turned cold, I was impressed by the extent to which I could choose a new water heater without thinking about air conditioners, telephones, windows, or practically anything else except the number of people in the house and the number of dollars in my bank account.

The title of Shore's book derives from the extended example he gives of trying to use his mother's recipe for Aunt Marti's Sachertorte. He illustrates the process of stepwise refinement as he moves from his mother's original instructions (prepare ingredients; bake cake) to a more detailed sequence of steps that are all within his own more limited repertoire of cooking operations.

Shore wants his readers to understand that programming computers is an exercise in careful and precise communication. The programmer must communicate with the machine, but even more importantly, the programmer must communicate with other human beings. In this sense, programming is a literary activity. When seen this way, some of the problems of the "software crisis" are easier to understand. Writing is easy, but writing well is hard.

Shore also wants us to look at programming as mathematics and as architecture. Here he laments the fact that millions of people are acquiring personal computers and programming them with the languages and techniques of the sixties. The concepts of abstract specification and information

hiding that were introduced in the early seventies are only beginning to be widely used today. Proofs of correctness, long recognized to be theoretically possible and highly desirable, are still no more than classroom exercises.

The tools of the writer, the mathematician, and the architect are the keys to managing the complexity of large software packages. The failure to manage complexity, in Shore's view, is the cause of our current software crisis. Shore does a good job of explaining and illustrating the problems of complexity and the failure of our tools in terms that should be accessible to readers of all backgrounds.

If you see this book in your local bookstore, buy a copy to give to a nontechnical friend. You might enjoy reading it yourself first.

Mathematica: A Practical Approach, Nancy Blachman (Prentice Hall, Englewood Cliffs, N.J., 1992, 380 pp., \$30)

I wanted to install 8 Mbytes of memory in my Macintosh so I could run Mathematica comfortably. Unfortunately, inadequate memory is not the only obstacle to using Mathematica. Mathematica is a large program with many capabilities, and before Blachman's book we had no adequately tutorial introduction to it. Stephen Wolfram, the creator of Mathematica, wrote a book that is supplied with the program, but as Blachman points out, learning Mathematica from that book is like learning English from a dictionary.

In my August 1991 column, I discussed some of the work Nancy Blachman and her company, Variable Symbols, have done to teach people to use Mathematica. She based the current book on undergraduate courses she taught at Stanford University. It draws on her extensive experience with the difficulties people encounter in learning Mathematica.

I'm impressed by the quality of this book. Blachman provided Prentice Hall

with camera-ready copy, so she can take credit for its attractive appearance and its excellent editing. Mathematica can also take some credit, since the book began as a Mathematica notebook, and many of the illustrations were originally done using Mathematica.

Learning Mathematica is not my top priority, so working my way through this book will be a background project for a while. In my next column I'll tell you how it's going. So far, I like Blachman's approach.

The Parents Guide to Educational Software, Marion Blank and Laura Berlin (Microsoft Press, Redmond, Wash., 1991, 424 pp., \$14.95)

The authors of this book are an educational psychologist and a developmental psychologist. Their intended readers are parents who want to give their children the advantages of computer-assisted education but don't know what to do next. As the authors point out,

Currently you can choose from over 10,000 programs, covering almost every subject imaginable. And, as always, excellence is rare. If you're like most parents, you probably don't see this array of options as a pool of resources but as a mire of confusion.

The authors have selected more than 200 programs for careful rating. They have applied several criteria to this selection, and some of these criteria differ from those used in selecting programs to be used in a school setting. Basically, all of the programs in the book are of high quality, attractive to children of the intended age group, sufficiently varied that children won't tire of them immediately, and inexpensive enough for many families to consider buying them. Where possible the authors tried to select programs that are helpful to the 10-15 percent of the

students who have learning difficulties in specific learning areas.

Most of the book is devoted to thorough reviews of the selected programs, but the authors begin with helpful introductory material. They explain how computers can help with learning. Then they review the major outlines of the elementary school curriculum, focusing on what skills are required by each area of study. Finally, they talk about how to set up an educational computer center at home and how to help your child use the programs that you acquire.

The reviews all follow a common format. The authors identify each program as being for a specific age range and school grade range. Then they summarize the hardware requirements, the abilities a child must have to use the program, and the curriculum areas that the program addresses. A detailed discussion of the program itself follows these short summaries.

I have no experience with any of the programs the authors have included, so I can't judge their reviews. However, the reviews that I read addressed issues that I'd want to consider in buying educational software. One interesting example was a \$14.95 program for children in the two-to-four age range. The child would not be able to start the program because of a complicated copy protection scheme, so the parent would always have to be there to start it. The authors emphasize this point without passing judgement. I admire their restraint.

I think this book is a really good investment for anyone considering buying educational software.

Annual phone list update

My year end would not be complete without the annual overhaul of my phone list. This year I finally automated the process.

Address Book Plus (Power Up Software Corp, 2929 Campus Drive, San Mateo, CA 94403, (415) 345-5900, \$99.95)

This program provides what most people need to handle their little black books. It lets you build a small database, define selection and sorting criteria, and generate reports in a variety of useful formats. These features are mostly built in, so that users don't need to use query or report generation languages.

The program supports printing formats corresponding to the popular pocket and desk appointment books. It also has a special Instabook format, which lets you print a stack of two-sided pages and turn them into a pocket-size address book, simply by cutting, folding, and stapling. I tried it, and it really works.

The program offers an integrated telephone dialing facility. It seems to have all of the flexibility necessary to deal with prefixes, area codes, choice of long-distance carrier, international dialing, modem control, and so on.

Power Up has gone to great lengths to make this program fit into your way of doing things. It can import or export files in formats that you define, and it has built-in formats allowing it to import from Hypercard and to export to personal organizers from Casio and Sharp. It also works with Power Up's mail merge program (Letter Writer Plus).

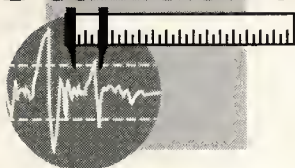
It's taken me a long time to get around to using a program like this, because no program comes close to doing all the little things I do by hand, but this one is good enough. What it does for me outweighs the things I'll have to give up.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 183 Medium 184 High 185

Micro Standards



Understanding the Acronym Tower of Babel

I am often asked what a particular acronym means, or where it came from. After reflecting on these questions, I realized that they might be the meat for an interesting column. After all, we use these "shortened" words to convey large amounts of information within a standard, or other technical documents.

Interestingly, many of the acronyms and definitions used today came about because of the telegraph and Teletype era. Since telegraphers had to tap in each letter of a word sent, they naturally wanted to use the smallest words possible. Consequently, a unique shorthand was created. This shorthand carried over to Teletype users, partly due to the slowness of the system and the difficulty of typing on the keyboard. Any of you that remember a KSR 35 will understand what I mean.

Interestingly, World War II and the Korean conflict also contributed to the acronym pool. The idea was to send as much information as fast as possible so the enemy wouldn't have a chance to locate the signal or decipher it.

Another interesting note concerns the contributions industry has made to the acronym Tower of Babel. The airline industry, in days before sophisticated reservation systems, used a form of shorthand called Fast Talk. This method let an agent in Chicago call an agent in Des Moines to reserve a seat through to Los Angeles.

Even with the advent of computer systems and better communications than Teletype, the airline industry developed a worldwide standard abbreviation system called SIPP (Standard Information Processing Protocols) codes. This system of codes allowed a reservations agent at United Airlines to send a great deal of easily understood information about a passenger to another carrier. For example, C-19 meant that the passenger required

special handling. Other codes indicated special food or an unpleasant passenger.

Similarly, the airlines also had a system for identifying lost baggage. For example, 02 Blue indicated a blue, hard-sided Samsonite bag and 03 denoted an American Tourister bag. Type 20, on the other hand, was a folding garment bag. I was one of the developers of United Airlines Total Apollo Baggage System (TABS) that made use of this information to automatically search for lost bags—a system that ultimately saved United several million dollars a year in mishandled baggage.

Of course, NASA (the US National Aeronautics and Space Administration) probably gets the prize for developing acronyms. It isn't unusual to pick up a NASA-developed document and find that it can't be read without the help of an acronym glossary close at hand. NASA's rationale is that so much information has to be conveyed that abbreviations, acronyms, and initialism (AA&I) enhance the readability of the document.

Because of limited space, I've taken two approaches in my acronym/definition list beginning on the next page. I've chosen some interesting acronyms and expounded upon them, and others I've just listed with their meaning. I could write volumes on this area alone or include multiple meanings, but I won't. I covered lots of ground with this column. You can probably guess, however, that I only touched the surface. If you have a favorite acronym that you would like to share and can tell a short (no more than two-paragraph) story, I'll include it in forthcoming Micro Standards columns.

Finally, some parting thoughts: Didja [sic] know that Soroc, the now-defunct spin-off terminal manufacturer from Lear Siegler, used an anagram for its name? Take a close look at Soroc, and you'll

continued on p. 72

Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 7-1230

warren@ssdunx.mdc.com

Glossary

Ack: acknowledgment, an old communications term that was used with Teletype systems. The telegrapher on the receiving end would indicate that a connection was made, or a message received, either by sending AK (later ACK) or their initials.

ADC: analog-to-digital conversion.

AIM: Association for Information and Image Management.

ANSI: American National Standards Institute, the governing body for the management and creation of standards (see IEEE, MSC, SPARC, and TCCM).

AT: Advanced Technology, an IBM-fostered term describing the basis of the PC bus architecture. This architecture built on the 8-bit and 16-bit PC and XT (Extended Technology) buses of IBM's earlier machines.

Ata: at attachment.

BER: bit error rate, an important metric when referring to storage or communications devices, which gives the number of bits at fault during various types of measures of the device.

BSR: Board of Standards Review.

Bsy: busy. Again, this is a term from telegraphy days. When queried, the downline telegrapher would respond with a busy reply, requesting a hold on traffic. Later with Teletypes, a busy signal was sent out much like the busy signal on your telephone being high, thus denoting busy, or setting of a "busy" bit in a UART, for example.

CAM: common access method/content access memory.

CBEMA: Computer Business Equipment Manufacturers Association.

CISC: complex instruction-set computer, a class of processor that has a large number of instructions to choose from. An example is the Intel

80386 or Motorola 68040. (See RISC).

CKD: count-key-data.

CR: connection resource.

CRC: cyclic redundancy check.

CS: continuous servo.

DADI: Directly Addressable Device Interface.

DASD: direct access storage device, a disk or other secondary storage device that permits access to a specific sector or block of data without first requiring the reading of the blocks that precede it.

DMA: direct memory access, transfer of data between the memory and a peripheral (or another memory) without intervention of the CPU (besides DMA controller initialization).

DOTS: digital optical tape system.

DTA: dedicated test article. Various test units may be used for various purposes in a test environment. A DTA is specifically dedicated to test purposes and as such may have special test ports and special power inputs, and the cabinets may or may not exist.

DUT: device under test. This refers to the actual hardware that is attached to test equipment (see UUT).

ECMA: European Computer Manufacturers Association.

EIA: Electronic Industries Association.

EISA: Extended Industry Standard Architecture.

Escon: Enterprise Systems Connection.

ESD: electrostatic discharge.

ESDI: Enhanced Small Disk Interface. This is the analog to SCSI. This interface, which was developed by Maxtor, provides a fast channel for disk drives. Originally, the "D" meant device, but the developers changed their aspirations and decided that a fast interface for disks was good enough. IBM did choose ESDI for early PS/2 models but is now moving exclusively to SCSI. This is one of the many interfaces that I. Dal Allan of ENDL Consulting, Saratoga, Calif., can take credit for.

FAT: file allocation table, a table that contains mappings of the physical locations of all of the clusters in all files on disk storage.

FBA: fixed-block architecture.

FC: fiber channel. This is technically an initialism and is a bad choice—writing out fiber channel makes more sense than using the initials.

fci: flux changes per inch.

FDDI: Fiber Distributed Data Interface, protocol description for optical networks and copper coaxial cable networks.

FEU: functional equivalent unit, a test unit equivalent to the actual finished device. As such, it has the same appearance as the finished article and operates in accordance with the functional requirements.

FFS: flat-file system.

FOM: figure of merit, another poorly thought-out use of acronyms. A figure of merit is a number that is determined from some weighting system that has preestablished boundaries.

fps: frames per second.

FPT: Forced Perfect Termination. This is a nifty trick developed by IBM to create a termination scheme that follows the changes in the impedance of a line. Theoretically, it implies an infinite length without attenuation of the signal. However, physics does get in the way, and the signal will die at some point.

FRAM: ferromagnetic memory.

Gbsi: gigabits per square inch.

HBA: host bus adapter. (This is SCSI talk.)

HIPPI: High-Performance Parallel Interface.

HIPPI-FP: Framing protocol.

HIPPI-LE: IEEE Std 802.2 link encapsulation.

HIPPI-PH: physical layer.

HSC: hierarchical storage controller.

Glossary (continued)

IDC: insulation displacement connectors.

IEC: International Electrotechnical Commission.

IEEE: Institute of Electrical and Electronics Engineers.

IIST: Institute for Information Storage Technology.

IOPS: input/output (requests) per second.

IPI: Intelligent Peripheral Interface, an interconnection standard that grew out of the Intelligent System Interface developed by ISS Sperry Univac. This interface is designed for large systems that have very high speed I/O channels.

JISC: Japan Industrial Standards Commission.

JTC1: Joint Technical Committee 1.

Lun: logical unit.

MCAV: modified constant angular velocity, a method used on compact disks and some read/write optical disks.

Mflops: millions of floating-point operations per second, a metric used to benchmark the overall processing capability of a microprocessor and math coprocessor.

MIG: metal-in-gap, a specialized read/write transducer used in high-performance disk drives.

MIPS: millions of instructions per second, a measure of a processor's horsepower. It is usually compared with a known system such as a Digital Equipment Corp. PDP 11/780 that has a MIPS rating of 1.

MO: magneto-optical, a technique of combining magnetic recording with optical recording to produce a high-capacity read/write device. Geoffrey Bate, a Santa Clara University fellow, pioneered this technique while at Verbatim Corp.

MR: magneto-resistive, a single-pole read/write transducer used in high-performance and high-bit

density disk drives.

MSC: Microcomputer Standards Committee, the governing body for bus standards.

MTBF: mean time between failures.

MTBS: mean time between stops. This has nothing to do with the rapid transit system in your town; rather, it is a measure used with rotating memory systems.

MTDL: mean time to data loss. This is my favorite acronym. Apparently, it describes what we all fear most (by the way, I saved my work at this point). That's the loss of all the work we have done up to the deadline because of a power failure, your cubical mate spills coffee down the grill work of your terminal, or the gnomes who control these things have decided it's your turn in the barrel.

MTSR: mean time to service repair.

MTTR: mean time to repair.

NIC: newly industrialized country. This just shows how silly we get with acronyms, but it is actually used.

NRE: nonrecurring engineering.

NRZ: nonreturn to zero, a recording method that uses the zero crossing as a reference point.

NSIC: National Storage Industry Consortium.

NWI: new work items.

OS, O/S: operating system.

OSF: Open Systems Foundation.

PCD: printed circuit disk.

POH: power-on hours.

PRML: partial response maximum likelihood.

Prej: port reject.

RISC: reduced instruction-set computer, a fast processor that has fewer instructions than a CISC and is defined as performing a register-to-memory move in one *t* (machine-cycle time).

QIC: quarter-inch cartridge; also the name of the tape committee, chaired by Raymond Freeman, Freeman Assoc., Santa Barbara, Calif., which developed the quarter-inch format and recording standards.

Rej: reject. Most people believe the general use of this term comes from the reject switch on phonograph record players that were marked "Rej" on the top of the knob.

RLL: run-length limited, which defines a code combining 1s and 0s that is a mathematical permutation allowing more information to be encoded on one transition. Notably, RLL is used in data recording and transmission.

R/W: read/write.

SCSI: Small Computer Systems Interface, one of the most important interfaces in the industry. SCSI grew out of the Shugart Associates System Interface (SASI). The now-defunct Shugart took the idea to ANSI under the guidance of I. Dal Allan (the father of modern-day interfaces), and the result was SCSI-I. Currently, SCSI-II—a robust 16-bit version with 32-bit capability—is expected to be deemed an ANSI standard soon.

SD3: project approval request (ANSI).

SPARC: Standards Planning and Requirements Committee, the group you address when preparing a new standard; also a microprocessor architecture.

SPOOL: simultaneous peripheral operations on line, when a high-speed device like a disk is interposed between a running program and a low-speed device such as a printer.

SSD: solid-state disk.

SWG: Special working group.

SSWG: Specific subject working group.

TAG: Technical Advisory Group.

TC: technical committee.

TCMM: Technical Committee on Microcomputers and Microprocessors.

TFH: thin-film head, a special read/write transducer that is made using semiconductor techniques. This type of head permits smaller geometries, thus more tracks per inch and greater

Glossary (continued)

density of storage.

TFM: thin-film media, a special recording media developed by using a sputtering process to lay down the thinnest possible layer of recording material to improve permeability and susceptibility to the recording process. This technique is used for optical recording, especially thermal-

magneto recording (TMR) that uses both optical and magnetic methods.

tpi: tracks per inch.

TPS: transaction processing system.

UART: Universal Asynchronous Receiver Transmitter.

ULP: Upper Layer Protocol. You can tell this came from technologists who deal with systems and storage devices. Communications people have their own jargon, and they would tend to point out

the layer in the seven-layer OSI model.

UUT: unit under test. (See DUT.) UUT is similar to a DUT depending on whom you talk to. Sometimes, UUT refers to software under test, while DUT refers primarily to hardware.

WORM: write-once, read-many, an optical device that laid the foundation for the optical systems being used today.

notice that it spells "Coors"—even their logo was the top of a beer can.

Remember NBI, the fast-rising word processor company of the seventies and early eighties? NBI stood for "nothing but initials."

And, one of the more interesting acronym histories belongs to *EDN* magazine. Most people erroneously think *EDN* stands for "Electronic Design News." About 40 years ago when *EDN*

started, the name meant "Electrical Design News." However, the magazine became generically known as *EDN*, and the publishers decided that would be its official name. But H. Victor Drumm, the publisher during the seventies and eighties, felt *EDN* stood for "everything a designer needs," which shows that an acronym never outshines brilliance.

Want to send me your acronym story? Mail it to me at the address shown

earlier, I'll be happy to receive it.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 180 Medium 181 High 182

Micro Law

continued from p. 6

prepare any response to your first.

It is the policy of Bit Bucket Consulting Services, Ltd. not to have its personnel enter into confidentiality relationships regarding pending litigation until the engineer and the company are sure they can appropriately support the position of the party for which the consulting/expert witness services are to be performed. I tried to convey this during our telephone conference on November 14, and thought that I did. But apparently I was not successful in properly communicating that position. I regret any confusion that may have been caused by failure of communication.

Accordingly, I have not read the materials enclosed in your letters and will not do so unless and until you advise me that I can do so without creating any mutual problems. I will study the two SIMM patents and any other

documents of public record that you feel I can appropriately examine in the light of the foregoing policy of Bit Bucket Consulting Services, Ltd. Then I will attempt to reach a preliminary decision whether I feel that the patents are valid and whether I can otherwise support the basis of your claim against Toshiba and NEC as to SIMMs infringing your two patents. I will check with our company records to make sure no conflict exists and will promptly get back to you.

If there is any problem with this, please advise promptly. Thank you for your interest in having us assist you.

*Sincerely yours,
Bit Bucket Consulting Services, Ltd.
By John Q. Kludge*

This is not perfect, but probably will work. For one thing, because you cannot claim not to have seen the enclosures, it may be argued that you must have read them. I don't know what you

can do about this, unless a lawyer or clean-room service reads and screens all your mail for you. I welcome suggested improvements from readers. In any event, those of you who want to consult or serve as expert witnesses now have an interesting new problem to worry about.

References

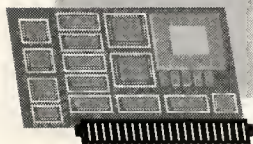
1. Civil Action No. 90-1477-A, E.D. Va., May 9, 1991, reprinted at 13 Comput. L. Rep. 1192 (Aug. 1991).

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 177 Medium 178 High 179

On the Edge



Firmware standards

[The P1275 working group is developing a proposed standard for boot firmware based on the machine-independent Open Boot firmware. Mitch Bradley discusses Open Boot's design and benefits of standardization.]

I invite readers to send information on a tool or method that solves problems, for consideration in future columns. —C.W.J

Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 7-1230

warren@ssdunx.mdc.com

Mitch Bradley

Sun Microsystems

Firmware is the ROM-based software that controls a computer between the time it is turned on and the time the primary operating system takes control of the machine. Firmware's responsibilities include testing and initializing the hardware, determining the hardware configuration, loading (or booting) the operating system, and providing interactive debugging facilities in case of faulty hardware or software.

Historically, firmware designs have been proprietary and often specific to a particular bus or instruction set architecture (ISA). This need not be the case. Firmware can be designed to be machine-independent and easily portable to different hardware. There is a strong analogy with operating systems in this respect. Prior to the advent of the portable Unix operating system in the mid-seventies, the prevailing wisdom was that operating systems must be heavily tuned to a particular computer system design and thus effectively proprietary to the vendor of that system.

Standardization

Standardizing firmware would offer several advantages to designers and users, including

- **Consistency across different systems.** It is easier to learn one firmware command set and use it across different systems from different vendors than to have to learn a different set of commands for every different system.
- **Avoiding duplication of effort.** Without a standard, every computer manufacturer must reinvent the firmware wheel. This effort is largely wasted. The availability of standard firmware offers manufacturers the option of buying proven firmware off the shelf, rather than designing and building it from scratch. Porting is almost always cheaper than writing from scratch. The existence of a standard also will raise the level at which value is added, allowing improvements to be made on top of a proven base, rather than continually redesigning the base.
- **Reducing design cycle times.** Designers would have one less thing to build for each new system architecture if standardized firmware is available.
- **Good, not half-hearted, firmware.** Since firmware is not visible to the average user and is rarely cited in marketing literature or measured in benchmarks or reviews, it is often treated as an afterthought or a necessary evil. Consequently, firmware design has not received the same level of attention as other, more visible software components. The requirements imposed on firmware by the desire to standardize it force us to take a hard look at many issues that might otherwise be swept under the rug.

Among the many firmware solutions available on different machines, Open Boot is the only one that has been proposed as a multivendor standard.

Open Boot advantages

We designed Open Boot firmware with open systems and standards in mind. We intended from the outset to port it to many different machines with widely varying bus structures, ISAs, and configurations.

Plug-in drivers. A key Open Boot feature is support for self-identifying devices. Consider a computer with an open expansion bus, such as VMEbus or SBus. An independent board vendor (that is, not a system manufacturer) of a card that plugs into the bus wants the system to recognize and use that card. In an operating system environment, this is easily accomplished. The board vendor supplies a driver on a diskette that can then be loaded onto a hard disk or installed into the operating system.

It is more difficult to support third-party devices in the firmware environment because firmware operates before the system is ready to read the disk. Since it is difficult to merge third-party drivers into existing system ROMs, it is better to store a driver in a ROM on the card for the plug-in device to which it applies. Others have taken this approach, but most existing firmware systems store the driver in ISA-dependent machine language binary code, and thus it only works on computer systems from a particular vendor.

Open Boot also uses the plug-in driver technique. But instead of storing those drivers in machine language, Open Boot uses FCode. FCode is a machine-independent, byte-coded intermediate language for the Forth programming language. It is based on a stack-oriented virtual machine that may be easily and efficiently implemented into any computer. FCode drivers are incrementally compiled into system RAM for later execution.

In addition to its use for firmware

device drivers, FCode also provides a descriptive capability. Plug-in device cards use it to report their characteristics to the firmware and system software. Such characteristics may include the device name, model, revision level, register locations, interrupt levels, supported features, and any other identification information that makes sense for the particular device. System software may use this information to automatically configure itself for correct operation with particular devices.

Interactive debuggers. Open Boot uses the same runtime system that executes FCode drivers as the basis for an interactive Forth language interpreter. This interpreter can be used as a programmable debugger, allowing developers, users, and service personnel to isolate system problems in the event of a failure.

Flexibility. We designed Open Boot for adaptability. Its notation and structure for naming particular devices is based on a hierarchical device tree that mimics the bus configuration and physical addressing of the machine on which it is implemented. This structure applies equally well to simple, single-bus desktop machines and to backroom servers with multiple processors and complicated hierarchies of interconnected buses. We designed the name space for individual device names so that allocating names requires no central authority. Companies can design their products without appealing to a master name arbiter.

The Open Boot command language is open-ended. In addition to the standard commands that are present on all implementations, an arbitrary number of new commands may be added at any time, even by the user. Such additional commands may provide access to system-specific features or may simply be customizations for the needs and tastes of individual users.

Maintainability. Field ROM upgrades can be expensive. Open Boot provides a self-patching facility that allows many types of firmware bugs

P1275 Open Boot working group

Mitch Bradley
Chair
2732 Katrina Way
Mountain View, CA 94040
phone: (415) 961-1302
fax: (415) 962-0927
or via e-mail: mitch.bradley@eng.sun.com

to be fixed without changing the system ROM. The same facility can be used to add additional firmware capabilities to systems in the field, without changing the ROMs.

Towards standardization

Sun Microsystems began developing Open Boot in 1987. We introduced version 1 with our Sparcstation 1 machines. Version 2, introduced with Sparcstation 2, corrected a number of deficiencies (we learned from our mistakes) and added several new features. We now use Open Boot on all of our current machines; approximately 500,000 units in the field employ it.

Sun has licensed its Open Boot implementation to several other computer manufacturers. Force Computers, a leading supplier of VMEbus products, intends to use Open Boot on future products across several processor families and buses. Sun Microsystems offers the source code for an implementation of Open Boot under a licensing arrangement.

Working group. The IEEE P1275 Open Boot Working Group is developing a firmware standard based on Open Boot. The group meets monthly at various locations. Membership is open to all interested parties. For more information, contact the author (see box).

Several IEEE bus standards are making provisions for Open Boot. The Futurebus+ standard includes a mechanism for using FCode ROMs for self-

identification. The VME-D draft document mentions a similar mechanism. SBus, under consideration for IEEE standardization, uses FCode.

For a copy of the P1275 draft specification contact the working group. The specification may be used and implemented without licenses or royalties.

Mitch Bradley is a senior staff engineer at Sun Microsystems. He has designed analog and digital hardware, written Unix device drivers and other software, and been a system troubleshooter. Most recently he worked on the design, implementation, and promotion of the Open Boot firmware. He owns Bradley Forthware, a small company specializing in Forth implementations for various computers.

Bradley earned a BE in electrical engineering, computer science, and math from Vanderbilt University and an MS in electrical engineering from Stanford University. He also studied for a year at Cambridge University on a Churchill Scholarship. He is a member of the IEEE Computer Society, the Forth Interest Group, and the ANSI Forth Standards Team.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Survey Card.

Low 192 Medium 193 High 194



Send information for inclusion in Micro News one month before cover date to Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

The limits of chip density

Ware Myers, Contributing Editor

Semiconductor density promises to increase for many decades to come. James D. Meindl of Rensselaer Polytechnic Institute predicted in an invited lecture to Supercomputing 91 in Albuquerque last November (J.D. Meindl, "Gigascale Integration (GSI) Technology," *Proc. Supercomputing 91*, IEEE Computer Society Press, Los Alamitos, Calif., pp. 534-538). His analysis points to what he calls "gigascale integration," or one billion transistors on a chip, by about the year 2000 with continued progress for some 30 years into the new century.

Meindl analyzed a hierarchy of limits on transistor density: fundamental limits set by the laws of physics; material limits set by the physical structure and chemical composition of the likely materials, silicon and gallium arsenide; device limits such as, in the case of silicon MOSFET devices, channel length, gate oxide thickness, and others; circuit limits such as the constraints on switching logic circuits; and system limits such as clock skew.

In addition, he considered the practical limits influenced by manufacturing technology and economics. Meindl asked, "How many components/transistors can we expect to fabricate in a single silicon chip that will prove to be useful, i.e., be economically viable, at some designated future time?" He quantified the practical limits in terms of three macrovariables: minimum feature size, the square root of the die area,

and the packing efficiency, that is, the number of transistors or components per minimum feature area.

From principles of physics such as the Heisenberg uncertainty principle, Meindl derived two fundamental limits which he plotted on a log-log power-time delay field. One side of these curves constitutes an impossible region, an area where the power or the time delay is less than that required to perform a switching operation.

In a similar manner he derived various other limits. He plotted limits on switching operations on the power-time delay field. Limits on transmission operations were plotted on a field of interconnection path length versus the corresponding signal propagation or response time. These lines gradually boxed in areas on the fields where operations are feasible.

Along the way he discovered that "on the basis of bulk material limits per se, silicon is not inferior to gallium arsenide as a material for gigascale integration." That is, as switching circuits get closer to the limits he hypothesizes, silicon will reassert itself over gallium arsenide's present electron-mobility advantage.

After considering a number of metrics and limits, Meindl felt that one figure of merit, in particular, was singularly instructive. The chip performance index, as he calls it, is the number of transistors on a chip, divided by the power-delay product. By power, he means the average power consumption during a binary switching transition of a logic gate. By delay,

he refers to the corresponding transition (or delay) time. Thus, as the number of transistors increases, the chip performance index also increases. And, as the power and/or delay time (in the denominator of the equation) becomes smaller, the chip performance index further increases.

This index increased by about 10^{13} from 1960 through 1990, Meindl calculated. He projects it to increase by another factor of 10^6 from 1990 through 2020.

Tiny lasers

Researchers at AT&T Bell Laboratories have made what they believe is the world's smallest semiconductor laser: about 5 microns in diameter. Seen through a scanning electron microscope, the lasers look like microscopic thumbtacks with the head of each tack 400 atoms thick.

The lasers operate in what is called a "whispering gallery" mode, so named after the sound effect noted in large cathedrals, where a whisper along the wall can be heard all along the inside perimeter. Like these whispers, photons travel with low losses around the edge of the laser. The lasers can be used as surface- or side-emitting devices. Each semiconductor disk laser is made of one or more layers of indium gallium arsenide sandwiched between two layers of indium gallium arsenide phosphide.



AT&T's micro disk laser

Nanotechnology

As technology becomes feasible on smaller and smaller scales, scientists continue to explore organic and inor-

ganic paths in search of building blocks for what may one day be computers that function at the molecular or atomic levels.

Natural nanocircuits. One biophysicist is studying proteins that conduct one-electron currents to find a model for protein-based nanocircuits—circuits made of organic materials 1,000 times smaller than those used today.

Jose Nelson Onuchic, an assistant professor of physics at the University of California, San Diego, believes it may be possible to design synthetic proteins that modulate minute currents, based on those found in vision, photosynthesis, and other biological functions. Onuchic believes an advantage to this line of study, as opposed to the traditional study of inorganic materials, is that nature has already evolved special proteins to perform the tasks computer designers are trying to emulate.

The goal of his research is the *biochip*, a molecular electron device that would include the protein or other organic chemical equivalents not only of wires but also of other computational gear, such as junctions, switches, resistors, and amplifiers.

Towards that goal, Onuchic and his team are pursuing three lines of research. The first is biochemical, in which researchers study the features of natural proteins that control the rate

of electron tunneling within and between the amino acids that form the links of protein chains. The electron transfer rate depends on the nature of the chemical bonds traversed by the tunneling electrons as they jump from atom to atom and by the geometry of the protein. Therefore, a second line of research explores the rules that govern the folding of natural proteins into their effective shapes.

The third line of research, which builds on the results of the other two lines, is directed toward making a working molecular electronic device. These efforts aim at creating a one-molecule memory element called a shift register, along with a one-molecule amplifier to read the information stored in the shift register. Onuchic believes it might also be possible for proteins used as memory and computing elements to self-assemble on biochip surfaces, as proteins do in living cells.

Molecular manufacturing. Alternatively, some researchers are working towards molecular manufacturing, the ability to build nanomachines one molecule or atom at a time. In contrast to contemporary microcircuits, in which billions of electrons flow from one place to another, nanometer-scale circuits might perform the same operation with the change in shape or position of a single molecule.

Micro bits

Amador Corporation, a Minnesota-based, conformity assessment testing laboratory, has announced a cooperative arrangement with the German VDE Testing and Certification Institute to test **US information technology equipment** for export to Germany.

IBM and the Center for Advanced Research in Biotechnology are developing a portable software system for **computational structural biology**, including programs to compute

protein structural data and model protein molecules.

Symbmath, an expert system that solves mathematic problems in symbolic formula or through numeric computation is available as shareware under the file name SM13A.ZIP from the directory MSDOS.Calculator in Simtel 20 at File Transfer Protocol sites. Symbmath requires significantly less RAM than most comparable software—640 Kbytes, as opposed to as much as 4 Mbytes.

To build such small devices, scientists must learn how to manipulate individual molecules or atoms. Current methods of molecular manipulation rely on the random motions of atoms to form the desired compounds. But molecular nanotechnology aims at moving individual atoms or molecules and snapping them into place.

Some progress has already been made. Scanning probe microscopes drag an ultra-fine stylus over a microscopic surface, thus mapping out its shape and allowing scientists to see individual atoms. Researchers have found that they can sometimes get atoms to stick to the microscope tip, enabling them to move the atoms around.

Using this technique, IBM researchers in San Jose in 1989 positioned 35 atoms of xenon to spell out their corporate logo. The achievement prompted others to create their own *nanoword*: scientists at Hitachi's Central Research Laboratory in Japan spelled out "Peace '91" by removing sulfur atoms from molybdenum disulfide, and a Stanford University student inscribed the first page of *A Tale of Two Cities* on a surface about the size of a red blood cell.

K. Eric Drexler, a visiting scholar at Stanford, is one of the founders of the Foresight Institute and the Institute of Molecular Manufacturing. He argues that manufacture at the molecular level is feasible and will one day produce nanomachines controlled by submicron, 1,000-MIPS CPUs and powered by nanomotors. Among the uses of such devices could be swimming through blood vessels to find and destroy viruses or bonding chlorine atoms from the atmosphere to protect the ozone layer. The machines would be manufactured by other nanomachines called molecular assemblers, tiny robot arms that position molecules precisely, bonding them into place in the design. (See related story on p. 7)

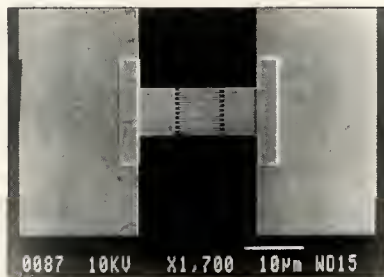
Alternative methods rely on the tendency of certain molecular components to stick to one another in the design of structures, thus enabling self-assem-

bling or self-replicating devices. Nadian Seeman, a chemistry professor at New York University, has created a design that allowed DNA strands to assemble themselves into a cubelike structure. Julius Reebek, Jr., a chemist at Massachusetts Institute of Technology, designed a synthetic molecule that serves as a template for two other molecules that combine to form a copy of the original.

1.2-ps photodetectors

A graduate student at the University of Michigan has used low-temperature-grown gallium arsenide to produce what she says may be the fastest light-detecting microchip. The device's 1.2-ps response time is six times faster than commercial photodetectors.

The chip's developer, Yi Chen, credits the special type of gallium arsenide, developed by researchers at MIT, for its speed. The material responds very quickly to light, creating a current that stops and starts instantly as a laser pulse starts and stops.



Chen's interdigitated electrodes

Chen's biggest technical challenge was developing an electrode structure to match the sensitivity of the gallium arsenide. She developed a way to use submicron electron-beam lithography to create interdigitated electrodes about 0.2 microns or 2,000 angstroms apart. By condensing the photodetector's electrodes into a smaller area, the design prevents the loss of signal-carrying electrons in the gaps between electrodes. The detector achieves an internal quantum efficiency of 68 percent (collecting 68 out of every 100

electrons) as opposed to traditional electrode structures that achieve about 1 percent.

According to Chen, the device holds promise for fiber-optic communication networks hundreds of times faster than current systems, precision 3D inspection systems, vehicle collision avoidance systems, and advanced medical imaging techniques. Chen is a graduate student in the school's applied physics program and a post-doctoral fellow at AT&T Bell Laboratories.

Cobol coinventor dies

Rear Admiral Grace Murray Hopper, known as "the first lady of software," died at her home in Arlington, Virginia, on January 4. She was 85.

Hopper was a pioneer computer programmer for the US Navy and coinventor of Cobol. She is credited with coining the term *bug* to describe the problems that plague computers and programs.

Admirers described Hopper as a vigorous, tireless, and occasionally contrary woman, with a healthy contempt for those unwilling to try new ideas. She once said, "The only phrase I've ever disliked is, 'We've always done it that way.'"

Hopper earned a PhD in mathematics from Yale University and taught at Vassar College before joining the Naval Reserve in 1943. After World War II she remained in the Naval Reserve and joined the company building the Univac I. The company later merged into Sperry, where she developed an idea that led to Cobol. She was the US's oldest active duty military officer when she retired in 1986.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 198 Medium 199 High 200



New Products

Send announcements of new microcomputer and microprocessor products to
Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

Joe Hootman

University of
North Dakota

Windows software

Speech from Windows

Using synthesized speech, Monologue for Windows reads aloud text, numbers, and data from Windows 3.0 and its applications. As in the manufacturer's original version, Monologue for Windows converts digital information into simulated speech in two phases using a set of phonetic translation and pronunciation rules. The software analyzes and translates text into sound descriptors, a phonetic language containing more than 1,000 rules to incorporate pitch, duration, and amplitude codes. Then it converts the language into speech signals. Algorithms drive speech tables that incorporate rules for merging signals into continuous speech.

Users can adjust the speed, pitch, and volume on screen. A dictionary manager lets users instruct the software how to pronounce words, abbreviations, acronyms, and symbols that may not comply with phonetic rules. The software requires 2 Mbytes of RAM, MS/PC-DOS 3.1 or higher, Windows 3.0 or higher, and a hard disk with at least 2 Mbytes available. *First Byte*; \$149.

Reader Service No. 10

Video window

X.TV is a video window that users can manipulate like any other window on workstations running X windows. Users can reposition X.TV anywhere on the screen, scale it to full screen, or reduce it to icon size. An on-screen control panel accesses audio-video functions, including volume, brightness, and contrast. A software push-button activates the frame grabber. Images can be named and saved to a disk file in raw data, TIFF, or Targa formats.

X.TV receives images through VMEbus, SCSI, or RS-232 port buses from the manufacturer's RGB/View hardware. RGB/View processes im-

ages independently from the CPU and receives from a variety of sources including video cameras, scanning electron microscopes, and infrared devices. *RGB*; \$750 (*X.TV*), from \$8,995 (*RGB/View*).

Reader Service No. 11



RGB's X.TV video window

VMEbus development kit

Programmers can develop Microsoft Windows 3.0 applications in a VMEbus environment using the XVME-984 Windows VMEbus Toolkit. The kit is a utilities package designed to run with the manufacturer's line of PC/AT VMEbus processors and support modules. It includes VMEbus Manager, a Windows 3.0 application that accesses and monitors the VMEbus. A library of functions contains the low-level code that configures and communicates with the manufacturer's VME I/O products. Demonstration programs are also included that offer examples of library uses. *Xycom*; \$650 (*with Windows*), \$500 (*toolkit only*).

Reader Service No. 12

PC-mainframe software

The Extra PC-to-mainframe software for Mi-

Parallel-processing DSP

Texas Instruments says its TMS320C40 is the first digital signal processor built specifically for parallel processing. One of the biggest challenges facing the chip's design team was how to debug a chain or array of multiple processors. Chief architect and program manager Ray Simar had seen customers purchase individual debuggers for each processor in a parallel system.

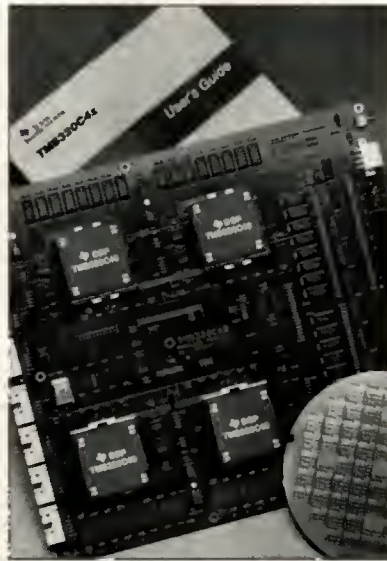
"Those debuggers cost \$10,000 to \$20,000 each," Simar said. "This gets prohibitive if you have 16 processors."

Space becomes a big problem, too. "They'd have nowhere to put them," he added, "so they'd have them hanging from the ceilings."

TI avoids rooms full of hanging debugger boxes by incorporating an analysis module onto each C40 chip. The module links through a JTAG (IEEE Std. 1149.1) port to debugger software, accessible in a windows-format interface. Users can monitor and test processors in the system individually or globally.

The analysis module is one of the key features of the C40, which Simar says is the first DSP built specifically for parallel processing. His team designed the chip from the ground up after looking at customers' difficulties incorporating their previous-generation DSP, the C30, into parallel processing systems.

According to Simar, customers linked the C30s together with multi-



Texas Instruments' TMS320C40 DSP

chip FIFO memories, an expensive process that also takes up space. "What we've done, essentially, is integrate 12 FIFOs onto the chip," he said. The C40 has six communication ports that link directly to other C40s with no external logic required. Thus, they link together tightly in pipelines, 2D arrays, or 3D arrays. This close proximity helps the C40 achieve what TI says is the highest performance of any floating-point processor: 275 MOPs with 320-Mbyte/s throughput per C40. Simar says the chips achieve even higher performance in parallel processing systems.

An on-chip, six-channel DMA coprocessor acts as a clerk for the on-chip

CPU. To allow the CPU to function smoothly, the DMA receives data, assembles it into packets, and stores it in a memory buffer (if necessary) until the CPU is ready for it. The C40 has an 8-Kbyte memory and two external memory buses to global and local memory. Through the interface, users choose how much on-chip memory space is allocated for the DMA and how much for the CPU.

Among the tools available from the manufacturer are an in-circuit emulator that provides parallel debug capabilities for embedded applications, a parallel-processing development system, an ANSI-compatible C compiler with parallel-processing runtime support library, an assembler and linker, and a state-accurate simulator. Third-party tools include the Multiprox code generation systems, an optimizing Ada compiler, and the Spox real-time operating system.

Simar says the C40 is suited for medical imaging (ultrasound, magnetic resonance, and CAT scans), pattern recognition for robotics, radar processing, 3D graphics, and military uses (image recognition and tracking). He also sees prospects for some nontraditional uses, including high-performance accelerators that attach directly to personal computers or workstations, high-bandwidth data transmission, and high-speed network interface. TI plans to ship in volume by mid 1992. *Texas Instruments.*

Reader Service No. 13

Microsoft Windows includes several easy-to-use features. Dynamic data exchange macros create automated information links between the mainframe and Windows applications. APL character support assists in financial and statistical analysis.

Extra also supports light pens, a feature designed primarily for the health

care industry. Command-line file transfer capability, for users more comfortable with DOS syntax, allows concurrent transfer of files using macros. A diagnostic trace facility allows users to record communications events on the network, to identify problems. *Attachmate; \$425. \$75 (upgrades).*

Reader Service No. 14

Cut and paste from X to MS

An X server integrates the X Window System in a Microsoft Windows environment. Software for Windows allows users to cut and paste between X and MS Windows applications and use MS Windows as a local window manager. It also features a complete on-line help system and several conveniences for

installing, configuring, and starting X applications from within the MS Windows environment.

Software for PC Unix, release 2.1, is an X server for 386/486-based machines running SCO or Interactive Unix. It adds support for Texas Instruments' 34020-based graphics accelerator board and a hot-key function allows users to switch to SCO's Multiscreen application. *Age*; \$495 (*Xoftware for Windows*), \$595 (*Xoftware for PC Unix*).

Reader Service No. 15

DOS-to-Unix connectivity

Aterm software allows two-way file transfer between systems and can emulate ANSI color terminals, graphics support, multiple screen display, and a menu-driven interface. It features a hot-key function that allows seamless movement between DOS and Unix systems. PC users can run DOS applications and switch immediately to applications running on the Unix host. Aterm is compatible with MS-DOS and Windows. *Specialix*; from \$125.

Reader Service No. 16

Signal-processing hardware and software

Background data collection

Easy Data collects data, modifies it, and sends it to the keyboard buffer without affecting normal keyboard functions. It imports data in the background while users work with another program in the foreground. Easy Data works with most software that receives data entry through the keyboard. Keyboard characters and macros can be inserted automatically before and after each data field to simulate the same keys that would be pressed in manual data entry. Data can also be selectively parsed so that only the required data transfers. *Labtronics*; \$145.

Reader Service No. 17

1,280-Mflops performance

Two of Texas Instruments' C40 DSP chips (see box on p. 79 and diagram below) are put to use in the Spirit-40 AT, an 80-Mflops DSP engine. Each 40-Mflops C40 includes 1 Mbyte of SRAM on the main bus, 1 Mbyte of SRAM on

the local bus, 64 Kbytes of boot EPROM, and memory expansion for up to 16 Mbytes of DRAM.

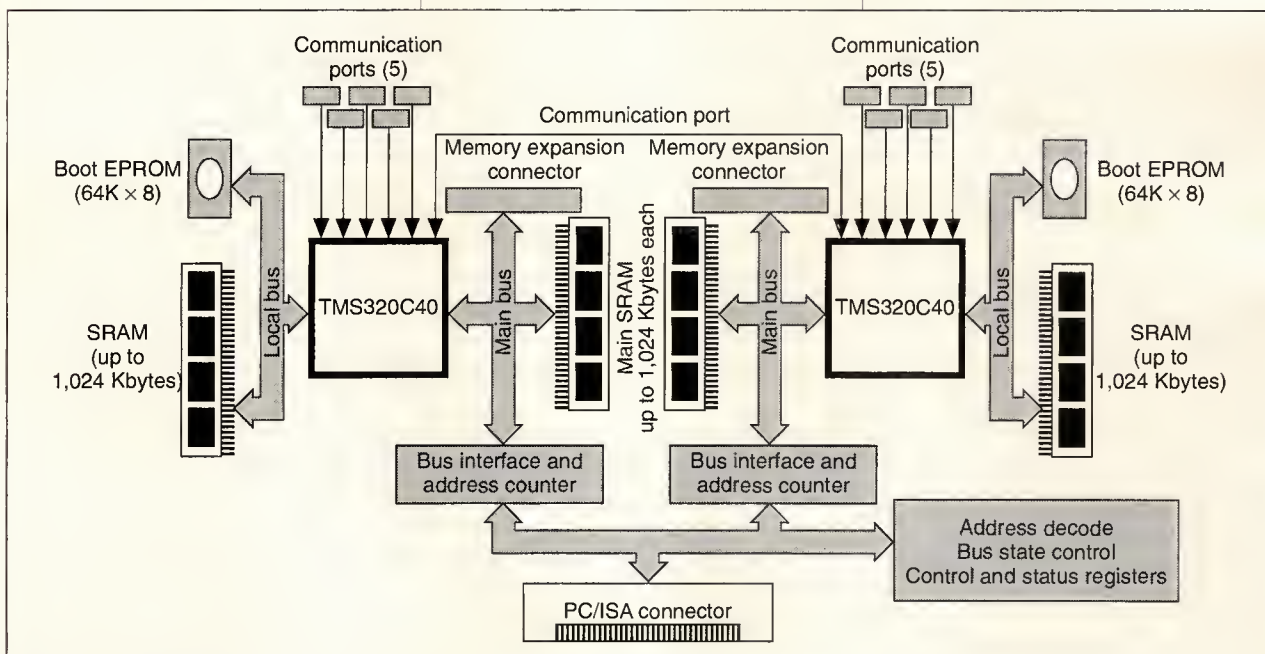
With the C40's six communication links, the Spirit-40 configures up to a six-dimensional hypercube engine with 64 nodes. The board occupies a 16-bit ISA slot and is compatible with 386- and 486-based machines. Up to eight boards fit in a passive backplane ISA system, yielding 640-Mflops performance. Two backplanes yield 1,280-Mflops peak performance. *Sonitech International*; \$8,995.

Reader Service No. 18

DSP package

Filter designers can analyze the performance of their designs on Monarch, a menu-driven DSP package with filter design, signal analysis, and graphical capabilities. Users can design finite impulse response filters (up to a maximum order of 512) or infinite impulse response filters.

The package includes Siglab, a DSP language with over 100 mathematical and system operations for performing



Sonitech's Spirit-40 AT



February 1992 issue (card void after August 1992)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest
(Add comments on the back)

Readers,
Indicate your interest in
articles and departments
by circling the appropriate number (shown on
the last page of articles
and departments):

150 151 152 180 181 182
153 154 155 183 184 185
156 157 158 186 187 188

159 160 161 189 190 191
162 163 164 192 193 194
165 166 167 195 196 197

168 169 170 198 199 200
171 172 173 201 202 203
174 175 176 204 205 206

177 178 179 207 208 209

Product Information
(Circle the numbers to receive
product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	



Does your library subscribe?

Readers: To recommend *IEEE Micro* for acquisition, complete this card and submit it to your librarian or department head.

Attention: Librarian/Department Head

I have examined *IEEE Micro* and would like to recommend
the magazine for acquisition.

Name (please print) _____

Department _____

Date _____

Signature _____

Sample copies are available from:

IEEE Computer Society
PO Box 3014
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1264

IEEE Micro
ISSN 0272-1732
Bimonthly: \$157/yr.

SUBSCRIBE TO IEEE MICRO

All the facts about today's chips and systems

☐ **YES**, sign me up!

If you are a member of the Computer Society or any other IEEE society,
pay the member rate of only \$23 for a year's subscription (six issues).

Society: _____

IEEE membership no: _____

Society members: Subscriptions are annualized. For orders submitted March through
August, pay half the full-year rate (\$11.50) for three bimonthly issues.

Full Signature _____ Date _____

Name _____

Street _____

City _____

State/Country _____ ZIP/Postal Code _____

☐ **YES**, sign me up!

If you are a member of ACM, ACS, BCS, IEE (UK), IEEE but not a
member of an IEEE society), IECEJ, IPSJ, NSPE, SCS, or other
professional society, pay the sister-society rate of only \$39 for a year's
subscription (six issues).

Organization: _____ Membership no: _____

☐ Payment enclosed Residents of CA, DC, Canada, and Belgium add applicable tax.

☐ Charge to ☐ Visa ☐ MasterCard ☐ American Express

Charge-card number _____

Expiration date _____

Month _____ Year _____

Prices valid through 12/31/92
02/92 MICRO

Charge orders also taken by phone:
(714) 821-8380 8 a.m. to 5 p.m. Pacific time
Circulation Dept.
10662 Los Vaqueros Circ., PO Box 3014
Los Alamitos, CA 90720-1264

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

Reviewers Needed. If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.

For reader service inquiries, see other side.

PO Box is for reader service cards only.

PLACE
POSTAGE
HERE

IEEE Micro

PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA



IEEE MICRO

IEEE Micro addresses an international audience of professionals who design and use microprocessors and microcomputers. In-depth articles examine microprocessor and board-level design, applications, system integration, VLSI, ASICs, fault tolerance, and hardware/software environments. Departments cover legal issues; industry and technical news from the US, Europe, and Japan; new products; standards; tools; and book and software reviews. *IEEE Micro* gives readers the information they need to stay competitive in this ever-expanding field.



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

IEEE COMPUTER SOCIETY

PO BOX 3014
LOS ALAMITOS CA 90720-9804
USA



signal and system analysis. Designers can create their own algorithms and DSP operations, which can be synthesized, tested, and saved for future use in other applications. Graphical displays feature overlay, zoom, grid, color, line style, and data location support. Monarch simulates real-world conditions, such as noise and fixed-point analysis. It requires MS-DOS 3.0 or higher, 640-Kbyte RAM, and a hard disk (or 2-Mbyte floppy). *Dynacomp*; \$549.95, \$10 (demonstration disk).

Reader Service No. 19

Communication hardware and software

Controls 255 nodes

COM20010 is a token-passing communications controller designed for high-speed intelligent data highways. The 2.5-Mbps device features a microcontroller interface and 1K × 8 on-board buffer RAM. It uses an Arcnet protocol engine and a token-passing protocol for real-time deterministic performance that supports data packet sizes from 4 to 512 bytes.

Up to 255 peer-to-peer nodes can network via the COM20010 at a data rate of 2.5 Mbps. System designers can implement peer-to-peer or master-slave communication schemes. The CMOS-fabricated device runs on a +5V power supply and comes in commercial and industrial temperature ranges. *Standard Microsystems*; \$11.36 (1,000s).

Reader Service No. 20

LANs connect to WANs

Local area networks connect into wide area networks with the LAN Transport Management System. LAN TMS, suited for Token Ring and Ethernet environments, eases interconnection between diverse LAN protocols, such as IBM Net BIOS, SNA, TCP/IP, Novell Netware, IBM server, and Banyan Vines. Synchronous data link control pass-through capability enables it to merge a non-LAN serial data stream with regular Token Ring traffic for trans-

mission over a common WAN link. Network administrators can monitor and troubleshoot problems on geographically dispersed LANs from a central management console. *General Datacomm*; from \$4,600.

Reader Service No. 21

Gateway software

Up to 30 Netware users can access transmission control protocol and internal protocol applications with Catapult gateway software. Its TCP/IP applications run over IPX, using Novell's Net BIOS. From the user's PC, Catapult's applications are routed to the OS/2 gateway, which replaces IPX with the TCP/IP transport protocols and sends the application packets to other hosts. The product runs on OS/2 PCs equipped with a network interface card supported by Novell's Netware Requester for OS/2 and over Ethernet, Token Ring, Arcnet, and Broadband networks running Netware 2.x or Netware 3.x. *Ipswitch*; \$2,975.

Reader Service No. 22

Sparcstations link to IBM

Two software products connect Sun Microsystems Sparcstations and IBM's system network architecture (SNA) over a Token Ring. Brx TR/SNA transparently supports all SNA services and supports local network management. Brx TR/IP leverages the transfer functions of the Token Ring network to include Unix systems. The technology enables traditional TCP/IP applications such as NFS, File Transfer, Mail, and Remote Login to operate seamlessly over Token Ring. Users can access remote Sparcstations as if they were locally connected to the same network. *Brixton Systems*; \$995 (both products and Token Ring card).

Reader Service No. 23

Special boards

Video interface card

The MZ-UT01 interface card establishes a video rate path between the

Scorpion real-time VGA frame grabber and Eighteen Eight Laboratories' PL2500 floating-point array processors. Multiple 640 × 480 × 8-bit images can be transferred in real time between the video card and the array processor. The half-length mezzanine card mounts directly on the PL2500 and draws power and ground through Span 32 bus interface connectors. The MZ-UT01 card comes with software to control the Scorpion board, interconnection cables, and a reference manual. *Univision Technologies*; \$2,495.

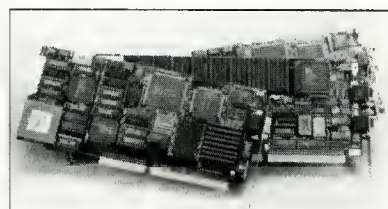
Reader Service No. 24

16.7 million colors

The Volante family of graphics processors offers three alternatives to PC users. The AT2000, compatible with PC ATs, displays up to 16.7 million colors at 640 × 480 resolution. At higher resolution (up to 1,024 × 768) it displays 32,768 colors. The MC1000, compatible with Micro Channel PCs, displays 256 colors at 1,024 × 768 resolution. The AT800, also compatible with PC ATs, displays 256 colors.

The processors, based on Texas Instruments' TMS34020, have a refresh rate of 72 Hz and a maximum bandwidth of 73 MHz. Their graphical interface works with MS Windows, TIGA, X Windows, and Nova Graphics CGI. *National Design*; \$1,295 (AT2000), \$995 (MC1000), \$795 (AT800).

Reader Service No. 25



National Design's AT2000, MC1000, AT800

Flicker-free screens

The Windows VGA 8800 graphics board boosts processing speeds for Windows 3.0 menus, windows, scroll-
continued on p. 83

64-bit RISC microprocessor

Mips has extended its RISC architecture to the 64-bit format with the R4000 microprocessor, a 1.3-million transistor chip designed to handle the growing volume of data found not only in large processors but in desktop machines as well.

According to Andy Keane, product development manager for the R4000, company designers looked in the direction RISC architecture was moving and decided to target the volume desktop market. "The early focus (of RISC) was on large machines, especially for databases," Keane said. "Now, RISC is more broadly defined." The R4000 is aimed at a range of applications, including high-end PCs, graphics workstations, database servers, and multiprocessing systems.

As users grapple with increasingly large amounts of data, the current generation of 32-bit machines, which handles up to 4 Gbytes of information, is becoming obsolete.

Based on observations that the

memory requirements of the average program grow by a factor of 1.5 to 2 each year, the company expects mainstream applications to exceed the capability of 32-bit machines by the mid-nineties. Doubling the number of bits in the microprocessor enables it to handle up to a terabyte of data.

Keane said his company is targeting PC users who are interested in the performance of a workstation but don't want to give up their investment in software. The R4000 is compatible with R3000 and R6000 applications, as well as Windows NT, Santa Cruz Operation's Desktop, and RISC/os, Mips' implementation of Unix. The chip performs in a 32-bit subset mode for most applications, employing its larger address only when necessary.

A 50-MHz clock drives the chip and operates a 100-MHz superpipeline. On-chip CPU components include a 64-bit integer processor, 64-bit floating-coprocessor, memory management unit, 8-Kbyte instruction cache, 8-Kbyte data cache, control and management facilities

for primary and secondary cache, and multiprocessing capabilities.

The R4000 comes in three versions.

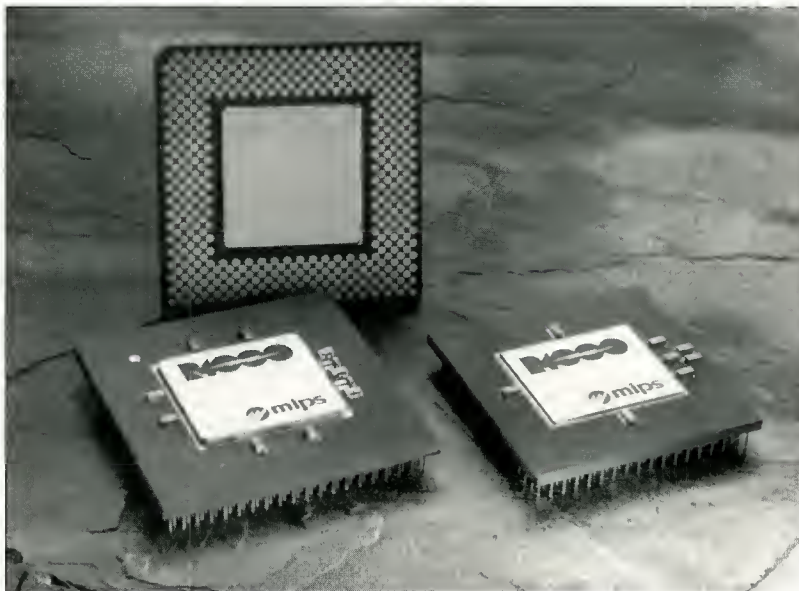
- R4000PC, which supports primary on-chip cache, is a 179-pin grid array package aimed at desktop, low-end servers, and embedded control systems.
- R4000SC, with secondary cache for uniprocessing applications, is offered in 447-PGA or land grid array packages intended for high-performance desktops and servers.
- R4000MC, with multiprocessing features and secondary cache, also comes in a 447-PGA or -LGA package.

Based on simulations of the SPEC benchmark suite, performance ranges from 40 Specmarks for the R4000PC to 60 Specmarks for the other two versions. According to the company, comparable Specmark performance has previously been achieved only from implementations of five to nine chips.

Among the development tools available are a C RISC compiler and a systems programmer package. The latter includes a cache memory simulator, an architecture simulator, and a development package.

Through the Advanced Computing Environment Initiative, a consortium of software, systems, and semiconductor companies that promotes an open computing environment, Mips has licensed six semiconductor manufacturers to produce the R4000. Last year Olivetti, Acer, and Mips previewed PC-size machines using the R4000. Mips expects commercially available machines using its microprocessor available in 1992. *Mips; \$700 to \$1,200, through licensed manufacturers.*

Reader Service No. 26



Mips' R4000PC, R4000SC, R4000MC

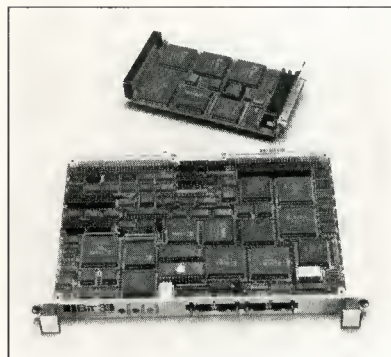
ing, and fonts up to 30 times faster than Super VGA standards, according to the manufacturer. A refresh rate of 70 Hz reduces screen flicker. Other features include 1 Mbyte of memory, 256 colors, and resolution up to 1,280 × 968 pixels. *Genoa Systems; \$495.*

Reader Service No. 27

CSBus-to-VMEbus link

Sun Sparcstations can interface with a VMEbus system with the Model 467 adapter. The adapter includes an SBus card and a 6U VME card, connected with a shielded cable. The systems communicate in two ways. Memory mapping permits the SBus and VMEbus host processor in one chassis to execute random access reads and writes to the destination bus as it would to a local memory. Alternatively, a built-in DMA controller permits transfers from the memory of one system to the other as fast as 20 Mbytes/s. *Bit 3 Computer Corp.; \$2,850.*

Reader Service No. 28



Bit 3's Model 467 adapter

Storage devices

Solid-state disk emulator

The Blue Flame III solid-state, non-volatile disk emulator boosts access speed by up to 20 times. The DOS-compatible card is an I/O mapped device built from 14 SIMMs. It features a 16-bit data path and transfers data up to 4 Mbytes/s. Capacities range from 2 to 56 Mbytes. Each card fits in a full-

length, 16-bit ISA bus slot. *Semi Disk Systems; from \$595.*

Reader Service No. 29

20-Mbyte Mac floppy

Quad Flextra is a very high density floppy-disk subsystem for the Macintosh with a 35-ms seek time and a data transfer rate of 1.25 Mbytes/s. Each 3.5-in. disk in the subsystem has a formatted capacity of 20.4 Mbytes, enough to hold font libraries, large graphics, and desktop publishing files. The external floppy subsystem measures 2.25 × 6.81 × 8.38-in., weighs less than 4 lbs., and has two SCSI connectors. The system includes a shielded cable, driver, utility software, and four disks. *Quadram; \$895, \$25 (additional disks).*

Reader Service No. 30

4.3-Gbyte tape drive

Two quarter-inch cartridge tape drives store up to 2.15 Gbytes of uncompressed data or 4.3 Gbytes of compressed data. The 9200 and the 9200C use a track density of 30 serpentine recording tracks and a record density of 67,733 bpi.

The drives support a synchronous burst transfer rate of 4.8 Mbytes/s and a sustained host transfer rate of 400 to 600 Kbytes/s for the 9200 and from 800 to 1,200 Kbytes/s for the 9200C. According to the manufacturer, each device accesses any file on a tape in two minutes or less and specifies a non-recoverable error rate of no more than 1 in 10¹⁵ bits. *Wangtek; \$900 (9200), \$1,100 (9200C) (OEM quantities.)*

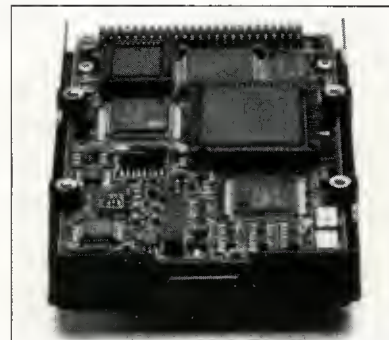
Reader Service No. 31

Winchesters for portables

Portable, laptop, and notebook computers can support more complex applications with the higher storage capacities of two Winchester disk drives. The MK-2024FC has a formatted capacity of 86 Mbytes and an average access time of 19 ms. The MK-2124FC holds 130 Mbytes and supports 17-ms access. Both drives come in a 0.75 × 2.5-in., 6-oz. package and

consume 1.8 watts when active and 0.15 W when inactive. Each drive has a 32-Kbyte cache memory and a 5-Mbyte/s data transfer rate. *Toshiba; \$495 (MK-2024FC) in (OEM quantities), \$695 (MK-2124FC).*

Reader Service No. 32



Toshiba's MK-2024FC

Laptop upgrades

Laptop Solutions upgrades hard disks for Toshiba laptops and notebooks, to store up to 120 Mbytes of data. Each upgrade includes installation and partitioning of the hard drive to user specifications, formatting of each logical drive, complete system diagnostics, and a one-year parts and labor warranty. The Houston-based company guarantees a 48-hour turnaround, including a 24-hour burn-in and test. *Laptop Solutions; from \$695.*

Reader Service No. 33

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 189 Medium 190 High 191

Product Summary

Joe Hootman

University of North Dakota

Manufacturer	Model	Comments	R.S.#
Chips			
Cirrus Logic	CL-GD6411 Graphics controller	A 3.3V, one-chip VGA controller supports prototyping and system development for notebook computers with 64 gray shades on a monochrome LCD. The device can directly drive a 512-color, active-matrix LCD. Features include simultaneous LCD and CRT displays, host bus interface logic, RAMDAC, and memory control logic. 160-pin quad flat pack; \$65 (100s).	80
Cybernetic Micro Systems	CY545B/CY500 controllers	The CMOS CY545B uses pulse and direction signals to generate up to 27,000 steps/s for full-, half-, quad-, and microstep applications. The CY500 provides programmable acceleration slopes for applications requiring 1 step/minute to 2,000 steps/s. Both accept ASCII or binary commands from a serial or parallel host. 44-pin PLCC or 40-pin DIP from \$25 (1,000s) (CY545); 40-pin LSI from \$10 (1,000s) (CY500).	81
Linear Technology	LTC1235 supervisor	Microprocessor supervisory circuit resets at 1V and offers a conditional battery backup feature for RAM data. Available in commercial or industrial grades and 16-pin SO or plastic DIP. \$3.85 (100s) (plastic, commercial).	82
LSI Logic	Sparkit-40/SS2 chip set	Manufacturers developing workstations compatible with the Sun Sparcstation 2 can sample the 40-MHz chip set, which includes the L64841 MMU, L64844 cache controller, and the L64846 DRAM controller. Manufactured for Sun and sold under license, the set works with three graphics controllers and Sunsoft software. \$844 (100s).	83
Philips Semiconductors	83C528 controller	This 80C51-compatible microcontroller features a watchdog timer, 32 Kbytes of ROM, and 512 bytes of on-chip RAM. The extra RAM allows the CMOS device to run compiled application programs in PL/M and C languages and provides space for context switching for stack enhancements in internal memory. DIPs, PLCCs, or quad flat packs; EPROM and one-time-programmable versions available.	84
Pletronics	Clock oscillators	Line of 50- to 120-MHz clock oscillators produces HCMOS/ACMOS-compatible output and drives 15 standard TTL loads. The crystal operates in fundamental mode at all frequencies. DIPs and mini DIPs with plastic/J and Gull-Wing leads; from \$3 (1,000s).	85
Xicor	X24C00 EEPROM	A 128-bit, CMOS serial device interfaces directly to a 2-wire serial bus and features software protocol that allows operation at a 1-MHz clock rate. 8-pin plastic DIP, type P, and plastic SOIC, type S; from 45 cents (1,000s).	86

Manufacturer	Model	Comments	R.S.#
Boards			
Data Translation	DT385 image processors	One-monitor, PC AT/EISA-compatible boards combine a frame grabber and graphics processor to acquire, digitize, and display standard and nonstandard video signals on a VGA monitor. An on-board TMS34020 processor accelerates Windows 3.0 graphics display; permits scaling and variable screen placement; and speeds arithmetic, convolutions, and morphological operations. From \$2,995, depending on memory.	87
Micro Express	Hi-Color Turbo VGA card	Designed around the Tseng Labs graphic chip set and the Sierra Semiconductor 15-bit DAC, the 32,768-color card supports 800 × 600 and 640 × 480 resolutions, as well as 1,024 × 768 resolution with 256 colors. A Turbo switch permits the card to switch between zero- and one-wait-state operation. \$185.	88
Newer Technology	fx/Overdrive accelerator	Variable-speed (40-/55-MHz) accelerator, when running at its fastest setting, promises to speed stock Macintosh IIx performance approximately 40 percent. When teamed with 16-Mbyte SIMMs, the surface mount device lets the IIx act as a workstation. The accelerator's motherboard installation leaves the Nubus and PDS slots open.	89
Parsytec	BBK-S4 Sbus/ transputer interface	Adapter board and accompanying software let Sparcstations and compatibles serve as a standard host to large-scale transputer systems for data transfers up to 8.8 Mbytes/s. The T225 and controller-equipped Sbus slave also assists image processing, real-time, and other computation-intensive applications and links with up to four external systems. \$3,950; quantity pricing available (10s).	90
Software			
Microware Systems	Polytron source code control	Utility package for automatic version control of application source code supports OS-9 and OS-9000 real-time operating systems. The set of 10 integrated modules supports multiuser development environments and features built-in file and user-access security. From \$895; available on disk or tape.	91
MIPS Computer Systems	R3000 Riscross tools	RISC microprocessor software development tools run on Sun-4/ Sun OS workstations and VAX/VMS systems. The cross-development tools include a K&R-compliant C compiler, System Programmer's Package, Cache 3000 memory simulator, and SPP/e development package. \$40,000 (set); from \$9,000 (individual prices).	92
Slate Corporation	Penbook	Electronic book Author and Reader software lets users translate, compress, and store Postscript documents and read them on a pen-based computer. Documents displayed as mixed text/graphics pages can be searched for user-defined words or phrases. A markup layer lets users annotate documents with personal notes. \$695 (Author); \$99 (Reader).	93

Product Summary

Manufacturer	Model	Comments	R.S.#
Peripherals			
HDS	Viewstation FX terminals	X Window, RISC-based, 14-, 16-, and 19-inch, 256-color terminals feature an Intel i960 CPU and two ASIC processors for communications and graphics integrated onto one board. Local Open Look and Motif window managers reduce network traffic and improve interactive performance. From \$2,799.	94
Mitsubishi	P-78U video printer	Monochrome autoscanning video printer promises to deliver 6 × 8-inch, A4-size prints with 1,280 horizontal dots per line in 256 shades of gray in 24 seconds. Composite, S-VHS, analog, TTL RGB, and Centronics parallel port input print in positive/negative, mirror-image, and multiformat print styles. \$2,999.	95
RGB Spectrum	1600U scan converter	Video scan converter with zoom, antialiasing, and 24-bit color processing changes high-resolution computer graphics to television format in real time. The unit automatically synchronizes to computer displays with 20/90-kHz horizontal scan rates. An optional RS-232 port lets users control all functions from a computer or ASCII terminal.	96
Miscellaneous			
Aristo Computers	Simcheck adapters	Memory tester family adds four adapters (ZIP memory chips; PLCC and SOJ memory chips; bank; and Apple Macintosh IIx SIMM). Basic Simcheck tests standard SIMM and SIP memory modules with 8 or 9 bits of 64K to 16 Mbytes. A two-line LCD shows instructions and test results. \$99 to \$345 (adapters), \$995 (Simcheck).	97
Lucas Duralith	LDC100 controller	Controller with internal EEPROM, 8-bit A/D converter, and serial interface lets designers test the applicability of touch-screen technology to their products by touching the screen at two opposite corners of the active display area. The controller is part of a development kit that includes a touch screen, two product development software disks, a user's guide, 220V to 110V transformer and plugs, and appropriate cables and connectors. \$695.	98
Philips Semiconductors	KM110BH/1x sensors	Hybrid magnetoresistive modules measure rotational speed down to zero using a toothed tachometer wheel with an inductive or Hall-effector sensor. Separations between tooth and sensor can be several millimeters, and tooth structure does not have to be defined closely. Samples available.	99
Solectek	Pocket fax modem	Portable, 9,600/4,800-bps send/receive modem supports DOS, Windows, and Macintosh applications. Users send faxes from within applications via a pop-up fax menu; they may continue working in the foreground or leave the application by entering a telephone number and normal printing commands. From \$299.95, depending on application.	100

Software Report

continued from p. 9

Micromachines will make it possible to create new medical equipment capable of performing diagnosis and treatment simply, accurately, and with less need for surgery. They will also contribute to the advancement of microsurgery techniques (eye surgery, suture of microscopic blood vessels, and so on) and the development of artificial organs to be placed inside the body. Biotechnologists will apply micromachines to cellular manipulation such as well separation, injections into cells, and cell fusion.

Announcements

NSF sponsors an active program for US scientists interested in working in Japan. As part of that program NSF prepared a directory of 150 Japanese companies that are willing to receive American researchers at their laboratories. The directory lists the company name, activity information, personnel, facilities, and research they will support, along with contact information. For copies of the directory, write to Japan Programs, Division of International Programs, National Science Foundation, Washington, DC 20550; fax (202) 357-5839.

The English summary of the outcome of the preliminary study on NIPT (or Sixth Generation Project) that I reported on in the April issue has been released. The Industrial Electronics Division of MITI published the 150-page "Report of the Research Committee on New Information Processing Technology" in March 1991.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 195 Medium 196 High 197

The Information Flood

Trying to manage the flood of information that passes before you can be a frustrating experience. Potentially useful information can be lost when you lack the means to organize and make sense of the multiple sources that arrive daily—as often as not, unbidden.

IEEE Micro's

On the Edge...

... offers a solution to this continuing problem.

In August, *On the Edge* begins a two-part tools discussion by James D. Gafford. The series will illustrate fairly simple ways for you to make use of sophisticated information management tools. The commercially available PC tools (MS-DOS or Macintosh) combine ease of use with information management power and flexibility. A common theme running through the series will be the creation and maintenance of a tool you can use to keep track of the information you read in *IEEE Micro* and other technical publications.

LOOK FOR THE AUGUST ISSUE of *IEEE Micro*

It will help you manage the information flood while gaining a better grasp of software tools and software issues in general.

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

Western Region: D. Rodney Brooks; Tel: (415) 905-0260; Fax: (415) 896-1512.

Eastern Region: Georgette Boone; Tel: (415) 905-0260; Fax: (415) 896-1512.

Recruitment and Classified Advertising: D. Rodney Brooks; Tel: (415) 905-0260; Fax: (415) 896-1512.

Director of Sales: Randall L. Stickrod, 544 Second St., Suite 200, San Francisco, CA 94107; Tel: (415) 905-0260; Fax: (415) 896-1512.

For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

IEEE MICRO, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

RS # Page #

Age	15	80
Aristo Computers	97	86
Attachmate	14	79
Bit 3 Computer Corp.	28	83
Brixton Systems	23	81
Cirrus Logic	80	84
Cybernetic Micro Systems	81	84
Data Translation	87	85
Dynacomp	19	81
First Byte	10	78
General Datacomm	21	81
Genoa Systems	27	83
GIGATEC SA	1	C.IV
HDS	94	86
Ipswitch	22	81
Labtronics	17	80
Laptop Solutions	33	83
Linear Technology	82	84
LSI Logic	83	84
Lucas Duralith	98	86
Micro Express	88	85
Microware Systems	91	85
MIPS Computer Systems	26, 92	82, 85
Mitsubishi	95	86
National Design	25	81
Newer Technology	89	85
Parsytec	90	85
Phase Three Logic	34	83
Philips Semiconductors	84, 99	84, 86
Pletronics	85	84
Quadram	30	83
RGB Spectrum	11, 96	78, 86
Semi Disk Systems	29	83
Slate Corp.	93	85
Solectek	100	86
Sonitech International	18	80
Specialix	16	80
Standard Microsystems	20	81
Texas Instruments	13	79
Toshiba	32	83
Univision Technologies	24	81
Wangtek	31	83
Xicor	86	84
Xycom	12	78

Coming Next Issue

The April Issue of *IEEE Micro* features articles selected from presentations at the third annual Hot Chips Symposium sponsored by the IEEE Computer Society's Technical Committee on Microprocessors and Microcomputers.

Don't miss
Hot Chips III
Read the April 1992 Issue of
IEEE MICRO

THE FOLLOWING INFORMATION IS AVAILABLE:

Contact the Publications Office; to facilitate handling, please request by number.

- Membership application, student #203, others #202
- Periodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Comppmail electronic mail brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures #193
- Student scholarship information #192
- Volunteer leaders/staff directory #196
- IEEE senior member grade application #204 (requires ten years practice and significant performance in five of those ten)

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

PURPOSE

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

MEMBERSHIP

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.



IEEE COMPUTER SOCIETY®

A member society of the
Institute of Electrical and Electronics Engineers, Inc.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and product reviews.

Periodicals. The society publishes seven magazines and five research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

OMBUDSMAN

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

EXECUTIVE COMMITTEE

President: Bruce D. Shriver*
17 Bethea Drive
Ossining, NY 10562
Phone: (914) 762-8030
Fax: (914) 941-9181

President-Elect: James H. Aylor*
Past President: Duncan H. Lawrie*

VP, Conferences and Tutorials: Barry W. Johnson (1st VP)*
VP, Educational Activities: Raymond E. Miller (2nd VP)*
VP, Membership Activities: Fiorenza C. Albert-Howard*
VP, Press Activities: Yale N. Patt*
VP, Publications: Harold S. Stone*
VP, Standards Activities: Gary Robinson†
VP, Technical Activities: Joseph Boykin†

Secretary: Ronald G. Hoelzeman*
Treasurer: Laurel V. Kaleda†
IEEE Division V Director: Bill D. Carroll†
IEEE Division VIII Director: Helen M. Wood†

Executive Director: T. Michael Elliott*

*voting member of the Board of Governors

†nonvoting member of the Board of Governors

BOARD OF GOVERNORS

Term Expiring 1992:

Alicja I. Ellis, Ronald G. Hoelzeman, Tadao Ichikawa,
C.V. Ramamoorthy, Sallie V. Sheppard,
Harold Stone, Akihiko Yamada

Term Expiring 1993:

Fiorenza Albert-Howard, Jon T. Butler, Michael C. Mulder,
Yale N. Patt, Anneliese von Mayrhauser,
Benjamin W. Wah, Ronald Waxman

Term Expiring 1994:

Mario R. Barbacci, Luis-Felipe Cabrera, Wolfgang K. Giloi,
Guyline M. Pollock, John P. Riganati, Ronald D. Williams,
Thomas W. Williams

Next Board Meeting

February 28, 1992, 8:30 a.m.
Cathedral Hill Hotel, San Francisco, CA

SENIOR STAFF

Executive Director: T. Michael Elliott
Publisher: H. True Seaborn
Director, Conferences and Tutorials: Anne Marie Kelly
Director, Finance and Information Services: Tod S. Heisler
Director, Board and Administrative Services: Violet S. Doan
Assistant to the Executive Director: Sandra K. Pfau

COMPUTER SOCIETY OFFICES

Headquarters Office

1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone: (202) 371-0101
Fax: (202) 728-9614

Publications Office

10662 Los Vaqueros Cir.
PO Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information:
(714) 821-8380
Publication Orders: (800) 272-6657
Fax: (714) 821-4010

European Office

13, Ave. de L'Aquilon
8-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05

Asian Office

Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 3408-3118
Fax: 81 (3) 3408-3553



IEEE OFFICERS

President: Merrill W. Buckley, Jr.
President Elect: Martha Sloan
Past President: Eric E. Sumner
Secretary: Karsten E. Drangeid
Treasurer: Theodore W. Hissey, Jr.

VP, Educational Activities: Edward A. Parrish
VP, Professional Activities: Arvid G. Larson
VP, Publication Activities: James T. Cain
VP, Regional Activities: Luis T. Gandia
VP, Standards Activities: Marco W. Migliaro
VP, Technical Activities: Fernando Aldana

COMM NEXUS

Presents

The LiSBUS™ Async I/O System

A solution of the 1990's for today's data transmission problems.



Outstandingly Simple and Reliable because LiSBUS™ is based on a breakthrough technology which uses the impedance of the bus cable to replace binary addresses. Consequently, data transmission management is greatly simplified and much more reliable than today's equivalent systems which require expensive software, hardware, and personnel investments.

Outstandingly Practical because it is easy to install and easy to operate. No special tools, workbench, or electronics expertise are needed. Anyone can be up and running in minutes. Just plug in the external modules and configure the system with the user-friendly LiSBUS™ Link Control Software. Each external module measures only around 2in. by 2in.

Outstandingly Flexible because a user can connect up to 60 peripherals or computers to a controlling computer through their RS-232C (COM) ports. To add peripherals, just extend the bus cable and add modules.

At an Unbeatable Price because at \$650* for the LiSBUS™ Starter Pack, no alternative can offer all these advantages

* specifications and prices subject to change without prior notification
Visa and MasterCard/EuroCard accepted. - CommNexus™ and LiSBUS™ are trademarks of GIGATEC SA.

combined into one product without spending a much higher amount. The **Starter Pack** includes all the user needs to connect four peripherals and a complete set of **LiSBUS™ Development Tools** to create custom applications.

LiSBUS™:

A product of our new CommNexus™ line of communication systems.

Special Pre-release Offer: you can receive the **LiSBUS™ Starter Pack** at the special pre-release price of \$550. Just call or mail in your order by February 29, 1992, at the latest. Deliveries will begin early March.

For more information and ordering contact:

In the USA and Canada:

In Europe:

Toll Free (800) 945-3002

Mon.-Fri. 9am - 9pm EST

GIGATEC (USA), Inc.
871 Islington Street
Portsmouth, NH 03801 USA
Tel. (603) 433-2227
Fax (603) 433-5552

GIGATEC SA
Ch. des Plans-Praz
1337 Vallorbe SWITZERLAND
Tel. 41 21 843 37 36
Fax 41 21 843 33 25

Reader Service Number 1

IEEE MICRO

APRIL 1992

Chips, Systems, Software, and Applications

Hot Chips III
Exploiting Parallelism for HIGHER PERFORMANCE



IEEE MICRO

Published by the IEEE Computer Society

April 1992

F E A T U R E S

- 8 Guest Editor's Introduction: Hot Chips III**
Norman P. Jouppi
-
- 10 The Mips R4000 Processor**
Sunil Mirapuri, Michael Woodacre, and Nader Vasseghi
Offering solutions for the increasing demands on address space
-
- 23 The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms**
William J. Dally, J.A. Stuart Fiske, John S. Keen, Richard A. Lethin, Michael D. Noakes, Peter R. Nuth, Roy E. Davison, and Gregory A. Fyler
Featuring efficient mechanisms for communication, synchronization, and naming
-
- 40 Organization of the Motorola 88110 Superscalar RISC Microprocessor**
Keith Diefendorff and Michael Allen
Achieving greater instruction-level parallelism in a second-generation RISC designed for use in low-cost personal computers and workstations
-
- 64 The Proposed SSBLT Standard Doubles the VME64 Transfer Rate**
Jack Regula
Adding a source-synchronous block transfer protocol to the VMEbus standard without affecting the backplane or electrical interface

Cover Image: Leo de Wys

Cover design: Design and Direction

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; IEEE Computer Society Headquarters, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription: \$23 in addition to IEEE Computer Society or any other IEEE society member dues; \$39 for members of other technical organizations. This journal is also available in microfiche form.

Postmaster: Send address changes and undelivered copies to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Second-class postage is paid at New York, NY, and at additional mailing offices. Canadian GST#125634188.

Copyright and reprint permissions: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of US Copyright Law for private use of patrons those post-1977 articles that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 29 Congress St., Salem, MA 01970. Instructors are permitted to photocopy isolated articles for noncommercial classroom use without fee. For other copying, reprint, or republication permission, write to Permissions Editor, *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Copyright © 1992 by the Institute of Electrical and Electronics Engineers, Inc. All rights reserved.

DEPARTMENTS

- 2 Mailbag**
- 3 Micro World**
European space shuttle
- 6 Micro Review**
Consciousness
- 72 Micro Standards**
PCMCIA notebook interface
- 74 Micro Law**
Game Genie
- 81 On the Edge**
Regression testability
- 85 Editorial Calendar**
- 86 Micro News**
Second-generation RISCs
- 89 New Products**
Devices, components,
software, signal-processing,
Windows software
- 94 Product Summary**

*Call for papers, p. 2, 9;
Membership application, p. 80;
Reader Interest/Service/
Subscription cards, p. 64A;
Advertiser/Product Index, p. 96;
Computer Society information,
cover 3*

IEEE Computer Society
PO Box 3014, Los Alamitos, CA 90720-1264
(714) 821-8380

Editorial: Unless otherwise stated, bylined articles and descriptions of products and services reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society. Send editorial correspondence to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. All submissions are subject to editing for style, clarity, and space considerations.

EDITOR-IN-CHIEF

Dante Del Corso
*Politecnico di Torino**

ASSOCIATE EDITOR-IN-CHIEFS

K.E. Grosspietsch
GMD

Ashis Khan
*Mips Computer Systems, Inc.***

EDITORIAL BOARD

John Crawford <i>Intel Corporation</i>	Gilles Privat <i>France Telecom</i>
Joe Hootman <i>University of North Dakota</i>	Ken Sakamura <i>University of Tokyo</i>
Victor K.L. Huang <i>National University of Singapore</i>	John L. Schmalzel <i>University of Texas at San Antonio</i>
David K. Kahaner <i>National Institute of Standards and Technology</i>	Arun K. Sood <i>George Mason University</i>
Hubert D. Kirmann <i>Asea Brown Boveri Research Center</i>	John W. Steadman <i>University of Wyoming</i>
Priscilla Lu <i>AT&T</i>	Richard H. Stern
Richard Mateosian	Philip Treleaven <i>University College London</i>
Teresa H. Meng <i>Stanford University</i>	Carl Warren <i>McDonnell Douglas Space Systems Co.</i>
Nadine E. Miner <i>Sandia National Laboratories</i>	Maurice Yunik <i>University of Manitoba</i>

MAGAZINE OPERATIONS

COMMITTEE

James J. Farrell, III (chair)
Valdis Berzins
Jon T. Butler
B. Chandrasekaran
Carl Chang
Manuel d'Abreu
Dante Del Corso
John A.N. Lee
Peter R. Wilson

PUBLICATIONS BOARD

Harold Stone (chair)
James J. Farrell, III
Ronald G. Hoelzeman
Mary Jane Irwin
Ming T. (Mike) Liu
Michael C. Mulder
Theo Pavlidis
Sallie V. Sheppard
Kishor Trivedi

STAFF

Marie English
Managing Editor
David Sims
Assistant Editor
H.T. Seaborn
Publisher
Marilyn Potes
Editorial Director
Pat Paulsen
Assistant to the Publisher
Jay Simpson
Art Director
Joseph Daigle
Production
Christina Champion
Membership/Circulation Manager
Heidi Rex,
Marian Tibayan
Advertising Coordinators

* Submit six copies of all articles and special-issue proposals to Dante Del Corso,
Dipartimento di Elettronica, Politecnico di Torino,
C.so Duca degli Abruzzi, 24, 10129 Torino, Italy; phone +39 11 556 4044;
Comppmail: d.delcorso; Internet: delcorso@polito.it; Bitnet: delcorso@itopoli
or

** Ashis Khan, Mips Computer Systems, Inc., 950 DeGuigne Drive, Sunnyvale, CA 94086;
Internet: ashis@mips.com.

Author guidelines for article submission appear in October 1992 *IEEE Micro*, pp. 28-29.
Or, contact Managing Editor Marie English at address at left.

Expedited Delivery

is available to all members residing outside the USA, Canada, and Mexico. We invite you to take advantage of this service providing delivery of your magazine weeks earlier.

For information on this service and its cost, contact:

**Expedited Delivery
IEEE Computer Society
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA
90720-1264 USA
Phone: (714) 821-8380
Fax: (714) 821-4010**

**IEEE Micro will reach
you faster.**

In the mailbag

(LK: liked; DLK: disliked; LTS: like to see)

April 1991

LK: Editorial [is] different and diffuse; DLK: digressive editorial; LTS: different and diffuse magazine. (K.T., Bushehr, Iran)

June 1991

LK: Datawave multiprocessor—probably many other applications would benefit having multiple CPUs mesh-connected on a single die; LTS: more on event-driven interconnection schemes. (J.C., Warsaw, Poland)

August 1991

LK: Strong content; timely, useful. LTS: ...SCSI-ESDI, mass storage. (C.R.A., Searingtown, NY)

LK: The article on parallel processing; [This is a hot theme; you will find others in future issues.—D.D.C.] LTS: graphic stations, image processors. [They are coming.—D.C.C.] (T.M., Tama-City, Japan)

LTS: Microprocessor-based systems (controller, data acquisition ...) (A., Makkah, Saudi Arabia)

LK: Guest Editor's Introduction by Ken Sakamura: USA vs. Japan—differences in design style. Congratulations to 68040 authors! (J.R.C., Warsaw)

October 1991

DLK: The artificial intelligence review; LTS: Mips R4000. [It is in this issue.—D.D.C.] (J.B., Bedford, MA)

LK: Micro Review, "Computers and creativity." (Y.S., Hino-City, Japan)

LK: Micro Law (W.S., Zurich, Switzerland)

LK: The magazine as a whole; LTS: more articles about bioengineering. (O.S., Rio de Janeiro, Brazil)

LK: Well-balanced technical information. (J.M.S., Florianopolis, Brazil)

LK: Micro Law [and the] Li and Chen [article]; DLK: Intel 860 article, already outdated, too much [like a] "technical handbook." [An article on the next-generation i860 is coming, but I do not think the current 860 is outdated!—D.D.C.] (T.S., Tromso, Norway)

LK: The idea of register windows; LTS: the specification of P1154 standard. (R.K., Osaka, Japan)

Call for Papers

**IEEE Micro seeks general interest
manuscripts for 1992 and 1993 issues.**

Suggested topics include biological computing, VHDL design and workstations, multiprocessing, optical computing, microcomputing to aid the handicapped, and ICs for HDTV. Submit six copies of manuscripts to Editor-in-Chief Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129 Torino, Italy (Internet: delcorso@polito.it), or Associate Editor-in-Chief Ashis Khan, Mips Computer Systems, Inc., 950 DeGuigne Drive, Sunnyvale, CA 94086 (Internet: ashis@mips.com).





Hubert Kirmann

Asea Brown Boveri
Research Center

CRBC, 1 CH-5405

Baden, Switzerland

Hermes, the European space shuttle

Manned space travel has long fueled human imagination, science fiction books, and heated debates about its utility and funding.

Visionary authors like Jules Verne were well aware of the latter problem. In his book, *From the Earth to the Moon* (1865), Verne envisioned the difficulty of multinational funding for a moon projectile. The German confederation was short of money and could only contribute 34,285 florins. For the Vatican, the idea came too soon, just after the rehabilitation of Copernicus in 1822. Switzerland only granted 257 Swiss francs, since the Swiss did not feel they could tie in any trade relationship by shooting a cannonball to the moon. The English did not participate at all, since they considered the project incompatible with their principle of nonintervention. France was a driving force behind the Gun Club project, but the largest part of the funding came from Russia and the United States.

The European program

The funding of the European manned space program in 1992 follows this 1865 scheme. The Gun Club's successor is the ESA, or European Space Agency. On the table of negotiation are three major projects involving manned space flights: the European *Hermes* space shuttle, the *Columbus* manned station, and the *Ariane 5* rocket. The ESA counts 13 members and two associates, but the main actors are the economically stronger countries of France, Germany, and Italy. England is absent. Switzerland accounts for 2 percent of the funding. The unit of negotiation is not the dollar, but the ECU (European counting unit). One ECU equals approximately US\$1.4.

The key component of the project is the *Hermes* shuttle,¹ which should carry a crew of

three and a payload of 3 tons to low orbit. Its development cost is estimated at 6,200 million ECUs. *Hermes* is primarily designed to serve the *Columbus* Man-Tended Free Flyer (MTFF) space station scheduled for the year 2001. It could also serve the US *Freedom* space station. The *Columbus* attached, pressurized module (scheduled for 1998), which is the European contribution to the *Freedom* international program, will dock to the *Freedom* space station. Docking to the Russian *MIR* space station is also foreseen. Estimates of *Columbus*'s cost are 3,700 million ECUs. (*Freedom*'s budget is 14,000 million ECUs, or US\$19.700 million.)

The *Columbus* precursor program foresees different *Spacelab* flights with the US space shuttle. This program involves auxiliary projects like the *Poem-1* polar orbit station or data relay satellites.

In a clever move, the ESA decided not to develop a special launcher for the *Hermes* shuttle, but to share the *Ariane 5* launcher with commercial satellites. Besides reducing development costs, this sharing produces two nice side effects. First, one can build on the experience accumulated with commercial satellites; second, the usefulness of the launcher does not come into question.

In fact, the unmanned European space program remains unchallenged. The Arianespace organization is responsible for the *Ariane* flights. *Ariane*'s clients include 90 percent of the world's satellite operators. They earned 20 million ECUs last year and received orders for 5,000 million ECUs. *Ariane*'s 100th stage just came off the assembly line. (It is the 50th rocket.) So, *Hermes* will be able to ride on that wave, although the launch capability of *Ariane 5* had to be boosted to put the 24.4 tons of *Hermes* in low orbit. The

Ariane 5 budget also increased to 3,500 million ECUs, but it seems secure now.

The ESA prepared a long-term plan, but the member states want to approve the yearly budget. Budget approval presents the main difficulty for the ESA—a similar situation to that of the US space station.² In November 1991 the ESA asked the member states to spend the impressive sum of 39,000 million ECUs and to make a yes or no decision on the *Hermes* program. To decrease the yearly budget, the ESA proposed to build only one shuttle, with the maiden flight postponed to the year 2002, one flight per year (instead of two), and operational capability by 2004. At their November 1991 meeting in Munich, the ministers preferred to adjourn the decision instead and await further studies.

The hesitation of the ministers just reflects the taxpayer's mood in the different countries. While 60 percent of the French favor manned space flights without reserve and consider it a matter of national pride, they would like the others to pay for it also. The Germans are more concerned with down-to-earth themes like environment and reunification costs, and the Italians simply lack the money.

But the usefulness of manned space flights is being questioned again—and not without reason. Indeed, the last 30 years since Yuri Gagarin's flight have shown that humans can contribute little in space. The Soviet *Progress* spacecraft demonstrated that automatic rendezvous in orbit was reliable, and the *Luna* probe brought moon rocks back to the earth at a fraction of the *Apollo* project's costs. Human flights are restricted to low earth orbit, but money is made on the geostationary orbit. And for astronomical or earth observation, nothing is as disturbing as a human being moving or sneezing inside the spacecraft.

But for the public, manned space travel is tied to emotion, pride, and dreams, and it is ready to pay for them. Despite the end of the Cold War, many

see space exploration as a competition, not unlike the America's Cup race. And this makes manned space travel probably the most important psychological motor to the development of spacecraft.

In 1992, six Europeans plan to enter space aboard foreign spaceships: three US shuttle flights (*Discovery* once and *Atlantis* twice) with *Spacelab* on board and two Russian missions to the *Mir* station. The real question is: Does Europe need its own shuttle when space tickets are already on sale?

***The usefulness
of manned space
flights is being
questioned
again—and not
without reason.***

And the Russians argue that the European shuttle already exists: They would like to sell their *Buran* (Blizzard) shuttle as an alternative to *Hermes*. This shuttle already performed its automatic maiden flight. *Buran's* next flight is scheduled for 1993, but budget restrictions and the political situation could delay it.

It is unlikely that the ESA will accept the Russian offer, because *Hermes* benefits are not found in operating it but in building it. The project allows the European industry to become acquainted with new space technologies. It would give work to 16,000 persons in the next 10 years.² It should form links between countries in another European project and possibly extend them to the Eastern countries. The Russians have already offered precursor flights on their simulators and training facilities. For industry and

universities, *Hermes* presents a challenge, a test bank, and an attraction pole for qualified engineers.

Is a space shuttle the correct answer to economic space flights? The fact that Russia also built a shuttle is no proof of its usefulness: In strategic games, one covers each move of the adversary. The US shuttle failed in at least two aspects. It was neither cheaper nor easier to operate than expendable rockets. And because the US shuttle must be manned, one failure delayed the whole US space program for two years. *Hermes* and *Buran* flights do not need to be manned, but a failure, even in the first unmanned mission, could stop the program as well.

The next generation is already on the drawing board: one- or two-stage spaceships that use air during atmospheric ascent and switch to rocket mode to reach orbit, like the British/Russian *Hotol*, the German *Saenger*, or the French *Star-H*. But the way to such spacecraft comes from mastering the technology, and this shall be worth the cost of *Hermes*. The fact that the ESA budgeted only one *Hermes* spaceplane shows that it is nothing more than a prototype.

Fault-tolerant multiprocessor

For the computer architects, the most challenging part of the *Hermes* project is its on-board computer, called the SEF.³ This fault-tolerant system, developed by Matra Marconi Space (Toulouse, France), will be the core of the *Hermes* avionics and support guidance, communication, and the mission itself.

The fault-tolerant computer system consists of a pool of four computers interconnected by serial links. It looks very similar to other avionics computers like the US space shuttle Primary Computer (1974), the SIFT (1978), or the FTMP (1978) computers. The architecture has not changed much in the last 17 years. Why should it? *La fonction fait l'objet.*

The most critical function supported by the SEF is the guidance, navigation, and control (GNC) of the spaceplane. GNC is normally a repetitive task: Read the input sensors, process the input data, and generate the command to the actuators. The GNC computer uses a high-performance RISC processor (the Sun Microsystems Sparc is a candidate) with 2 Mbytes of memory to provide 4 MIPS of processing power. The boards of the GNC computer interconnect via a Nubus (IEEE Std. 1196).

The main input sensors are the inertial navigation system, the global positioning system, and the radio altimeter. A set of sensors connects to each of the four GNC computers through a dedicated bus. The critical communication link to the *Ariane 5* system is duplicated.

To cover the time-critical flight phases, the computer masks errors rather than using time-consuming recovery methods. To this effect, four processors execute the GNC algorithms in parallel. The processors are synchronized to operate on the same input data set to ensure that they do not diverge. Each computer reads its inputs and sends their values to the other three computers over the 7-Mbps serial interprocessor link. The computers reach a consensus on the input data, process the data, and broadcast the result, so as to reach a consensus on the output value. Only then is the value forwarded to the actuators.

To offload the application processors from synchronization and match-

ing, a dedicated processor, called Data Manager, handles the four serial interprocessor links between the computers. These links, called the Interprocessor Network, provide a reliable clock synchronization with a 20-ms period. This new approach responded to recognition that synchronization is a critical and time-consuming function, especially when executing Byzantine agreements.

This arrangement can still fail because of common-mode errors. The most obvious is that the same software error may affect all computers. Therefore, the programs running in the different processors may be diversified, for example, written by different persons. So it becomes unlikely that the same error will affect all computers. This technique is called *N*-version programming. It is used today, for instance, in the *Airbus 320* computers.

This is also a new approach for another reason. Previous projects, such as the US space shuttle, did not foresee software diversity or let it execute on a distinct computing system. A representative prototype based on functional models of this fault-tolerant computer pool architecture has already been developed by Matra Marconi to validate the concept. Final space qualification will be the real challenge of the SEF development, especially with respect to the tools and methods involved. Too often, fault-tolerant computers have been a pill in search of a disease. The SEF offers a unique opportunity to apply the theory of fault-

tolerant computing where it is really needed. This is one of the merits of the *Hermes* project.

What is the future of *Hermes*? The Greeks named Hermes the god of eloquence, trade, and thieves. The future will show which name really applies, if not all three.

Acknowledgment

My thanks go to A. Blanc for his collaboration.

References

1. "Europeans Facing Major Hurdle in Implementing Long-Term Space Plan," *Aviation Week*, June 17, 1991, pp. 96-99.
2. J. Feustel-Buechl, "Wings for European Spaceflight—the Future of Space Transportation," European Space Agency, Paris, 1991, pp. 21-28.
3. A. Blanc and A. Mosnier, *Proc. AIAA/NASA Second Int'l Symp. Space Data Systems*, AIAA-90-5028, 1990.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 186 Medium 187 High 188

IEEE **MICRO**

**Are you reading
someone else's
copy?**

If so, why not apply for your personal subscription? Just complete the subscription portion of the Reader Service Card in this issue and drop it in the mail. Then you'll never have to wait to read *IEEE Micro* again.

Micro Review

Richard Mateosian

2919 Forest Avenue

Berkeley, CA

94705-1310

(510) 540-7745

Consciousness

Regular readers of this column have seen several reviews of books about human mental processes. In October 1988 I talked about Johnson-Laird's frustrating *The Computer and the Mind*. Last August I reviewed Penrose's *The Emperor's New Mind*, a work whose main conclusion about the inability of artificial intelligence (AI) to replicate human behavior depends on an as yet undiscovered theory of quantum gravitation. Then last October I looked at Boden's *The Creative Mind*, a work that takes a contrasting view of AI and fails to disagree completely with Penrose only because of a last-minute appeal to human chauvinism.

This time I have looked at a work that attempts no less a task than the complete explanation of consciousness. It is a serious and scholarly exploration of the great central problem of psychology and philosophy, the mind-body problem. I predict that many people will buy it, but few will read it, and fewer still will understand and adopt the viewpoint that it teaches.

Consciousness Explained, Daniel C. Dennett (Little, Brown, Boston, 1991, 524 pp., \$27.95)

The first thing I have to tell you about this book is that its explanation of consciousness, by the author's own admission, is sketchy. Dennett tries to explain the most important physiological and psychological facts and to answer the most widely known philosophical arguments that contradict his point of view. At many points, however, he simply gives the shape of the theory, leaving to further research the fleshing out of details. As he says in his appendix for scientists, in which he proposes several experiments,

Since as a philosopher I've tried to keep my model as general and noncommittal as

possible, if I've done my job right, these experiments should help settle only *how strong* a version of my model is confirmed; if the model were entirely disconfirmed, I would be well and truly refuted and embarrassed.

Dennett is a wonderful storyteller. His ability to make points and cut through jargon with well-chosen analogies, anecdotes, and caricatures is breathtaking. Again and again as the going gets rough, he finds a way to bring the discourse back to an arena in which the reader feels at home. For example, in discussing color vision, a favorite topic in philosophical discourses on human mental processes, he mentions the Rosenbergs' torn Jell-O boxes. Two spies could identify themselves to each other by producing the torn halves of a Jell-O box. Neither half's pattern has any intrinsic significance, but each matches the other perfectly. This story cuts through philosophical jargon that goes back hundreds of years and makes clear immediately Dennett's view of why there are colors.

In *The Selfish Gene* (Oxford, 1976), Richard Dawkins coined the term *meme* to describe complex idea units (like wheel, alphabet, calculus, the *Odyssey*, the theme from the slow movement of Beethoven's Seventh Symphony). Memes are central to Dennett's view of consciousness. He says,

Human consciousness is *itself* a huge complex of memes (or more exactly, meme-effects in brains) that can best be understood as the operation of a "von Neumannesque" virtual machine implemented in the parallel architecture of a brain that was not designed for any such

activities. The powers of this virtual machine vastly enhance the underlying powers of the organic hardware on which it runs. But at the same time many of its most curious features, and especially its limitations, can be explained as the by-products of the *kludges* that make possible this curious but effective reuse of an existing organ for novel purposes.

This certainly sounds like strong AI, the doctrine so vehemently opposed by Penrose. If you allow this definition to stand, you have to accept conscious machines or, alternatively, Dennett's assertion that we're all zombies. That is, Dennett believes there is no consciousness of the mysterious (epiphenomenal) sort posited by people who say that human beings must be more than "mere" Turing machines, no matter how closely machines can simulate their behavior. These are views, says Dennett, that do not deserve to be discussed with a straight face.

One of the most important things to learn from Dennett's book is how to apply his highly counterintuitive point of view. Most people who introspect about their minds tend to picture a control room in which a self gathers together sensory inputs and remembered information. The self uses these to make decisions and issue commands to the mechanisms that control speech, movement, and so forth. For example, in speaking of the effect of the "blind spot" where the optic nerve passes through the retina, many writers say that the brain fills in the missing part of the field of view. This view makes no sense unless there is an internal projection of the visual field and an internal observer viewing that projection in the control room. Descartes placed the control room in the pineal gland, but no serious modern thinker believes the control room model corresponds to actual brain function.

I don't want to give a detailed ex-

planation of Dennett's replacement for this model. To me, his view of a person is like a roomful of monkeys at typewriters putting out a single newsletter. He regards the self as a construct, like center of gravity, whose usefulness breaks down if you get too close. In fact, he uses the term "narrative center of gravity."

While most thinkers reject the Cartesian control room model, many let it enter implicitly into their arguments, especially in "thought experiments." Thought experiments are parables, like Searle's Chinese Room (see *Micro Review*, August 1991). The philosopher devising the thought experiment asks you to imagine a situation that is possible in principle but usually impossible in practice. Then you are asked to follow a handwaving argument that leads to the point the philosopher is trying to make. Dennett ridicules a few notorious thought experiments. These exercises are good examples of the application of his model.

This review of Dennett's densely packed 500 pages is necessarily sketchy and incomplete, and it may not give much inkling of the excitement I felt while reading it. If you are interested in this subject, you should invest the time necessary to read and understand Dennett's book.

Macintosh utilities

Every year after the MacWorld Expo in San Francisco in January, I receive a large number of Macintosh programs to review. This year there seemed to be a better selection of utility programs than I've seen in previous years.

Now Utilities, Version 3.0 (Now Software, 520 S.W. Harrison St., Suite 435, Portland, OR 97201; (503) 274-2800, \$129)

The Now Utilities is a package of 10 programs. The company has tried to cover all the bases, but other manufacturers provide better products for some of the functions. I think the best parts of the package are the Now

Menus and Super Boomerang.

Now Menus allows cascading up to five levels. This is ideal for use with the Apple menu under System 7, since the most natural way to organize Apple menu items is in nested folders. Super Boomerang remembers applications and documents that you have used recently and makes them instantly available by slightly modifying the operation of the file-selection dialogs used by all Macintosh application programs.

WYSIWYG Menus is another useful program. It causes each entry in a font-selection menu to appear in characters from the font named by the entry. Of course, this program has a few drawbacks. For example, the names of fonts like Symbol or Zapf Dingbats are rendered in greek or in meaningless pictures.

Startup Manager lets you determine which startup programs to use and in which order. This program can be very helpful in debugging startup conflicts.

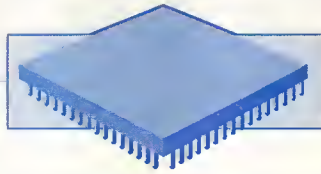
The other programs of the Now Utilities are also useful, but you don't have to use them. Each of the utilities can be installed separately. Even if you only use a few of them, you'll still get your money's worth.

Alsoft Power Utilities (Alsoft, Inc., PO Box 927, Spring, TX 77383-0927; (713) 353-4090, \$129)

Alsoft's package of seven utilities is more narrowly focused than Now's. Four of the utilities pertain to disk operation. The others are a menu utility that is similar to Now Menus, a screen dimmer, and the partially obsolete (for System 7 users) Master Juggler.

Disk Express II keeps the files of your hard disk stored as efficiently as possible. It reorganizes files on demand or once per day in the background. It also removes fragmentation and keeps frequently used files together. It performs its job one file at a time, so that it is interruptible and little damage is done if it crashes.

continued on p. 79



Guest Editor's Introduction

Hot Chips III

Norman P. Jouppi

Digital Equipment
Corporation

The annual Hot Chips Symposium presents the most current and exciting chip developments, as well as work in progress. The recent third meeting again boasted record attendance and required moving to the largest auditorium at Stanford University. The authors of seven of the most interesting and technically solid presentations were invited to submit papers for this special issue of *IEEE Micro*. Six authors agreed to submit papers, three were able to, and two appear here. In addition, this issue also carries an article detailing a recently developed and indisputably "hot" chip that was not presented at the symposium.

A theme running through the three special issue articles is that of exploiting parallelism for higher performance. Each product exploits parallelism in a different way.

Authors of the first article explain how the Mips R4000 exploits instruction-level parallelism through superpipelining. Superpipelining refers to the further pipelining of what are normally fundamental single-cycle operations in a pipelined machine. For example, on-chip cache access usually occurs in one cycle in pipelined microprocessors (not counting the usual cycle for address calculation). In contrast, the R4000 cache access is pipelined into three stages: two for cache access and one for tag comparison and control. The R4000 also provides support for a moderate degree of multiprocessing.

Next is a message-driven processor used in the J-machine at MIT. The message-driven processor

exploits parallelism through large-scale and fine-grain multiprocessing. Each processor can deliver a message and dispatch a task to handle it with a latency of under two microseconds. In comparison, this is within an order of magnitude of the time required for a main memory access in most computers. The architecture and hardware design of the J-machine supports up to 4,096 processors!

The third article describes the 88110 from Motorola. This microprocessor makes use of instruction-level parallelism by issuing multiple independent instructions in the same cycle (that is, a superscalar approach). The 88110 adds graphics support and a floating-point register file to the 88000 architecture. Many organizational features add to performance, such as the out-of-order issue of stores, nonblocking caches, and 10 independent functional units. All but one of the functional units can begin a new operation each cycle. The 88110 also provides support for a moderate degree of multiprocessing.

Hot Chips IV is already in the planning stages. It takes place August 9-11, 1992, at Stanford University. If you'd like to submit a presentation, contact Bob Miller at (510) 642-6037 (bmiller@ginger.berkeley.edu). If you'd like more information about the 1992 symposium, contact Glen Langdon at (408) 459-2212 (langdon@cse.ucsc.edu).

I thank those reviewers who helped referee submissions with an amazing one- or two-week turnaround, and all the other people who helped with this issue. □



Norman P. Jouppi is a member of the research staff at Digital Equipment Corporation's Western Research Lab in Palo Alto, California, and a consulting assistant professor in the Electrical Engineering Department of Stanford University.

Previously, he was one of the principal architects and implementers of the Stanford Mips processor. While at Western Research Lab he was the principal architect and implementer of the Multi Titan CPU. His current research interests include computer architecture, VLSI design, and VLSI CAD tools.

Jouppi received BSEE and MSEE degrees from Northwestern University, Evanston, Illinois, and the PhD degree in electrical engineering from Stanford University. He was a National Science Foundation Fellow from 1980 to 1983 and is a member of the IEEE, Computer Society, and ACM.

Readers with questions concerning this special issue may contact Jouppi at Digital Equipment Corporation, 250 University Ave., Palo Alto, CA 94301; jouppi@pa.dec.com.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

High 150

Medium 151

High 152

Call for Articles

Advanced Packaging and Interconnect Technology

April 1993 Special Issue

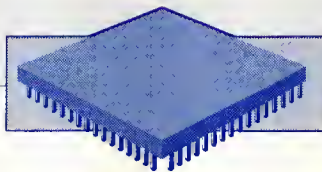
IEEE Micro

The April 1993 special issue of *IEEE Micro* will feature articles on advanced packaging and interconnect technology, as a companion issue to the April 1993 *Computer* special issue on multichip modules. The guest editor solicits manuscripts in the areas of

- critical packaging trends and issues;
- substrate and package technologies—for example, flexible, glass, or diamond substrates, few-chip packaging, or 3D packaging;
- attachment, bonding, and connection technologies, including fine-pitch surface mount, laser applications, known-good die, and interconnection trade-off analysis;
- system-level issues such as test, performance modeling, cooling, and EMI; and
- materials technology.

Authors should submit six copies of an original manuscript **by July 1, 1992**; notification of decisions is set for October 1, 1992; and the deadline for submission of the final version of each manuscript is December 1, 1992. For author guidelines, contact Clair Azada, Computer Society West Coast Office, (714) 821-8380.

Direct submissions and questions to Guest Editor David Misunas, MCC, 12100 Technology Blvd., Austin, TX 78727, phone (512) 250-3045, fax (512) 250-3045.



The Mips R4000 Processor

Computer architects estimate that the current generation of 32-bit machines will be obsolete by 1997. The R4000 employs a 64-bit architecture, using 64-bit registers and generating 64-bit virtual addresses. Superpipelining techniques allow it to process more instructions simultaneously than the previous generation of microprocessors. Specmark ratings indicate it performs higher than other single-chip microprocessors.

Sunil Mirapuri

Michael Woodacre

Nader Vasseghi

Mips Computer Systems

The R4000 is a highly integrated, 64-bit RISC microprocessor that provides a simple solution to the increasing demands on the size of address space, while maintaining full compatibility with previous Mips processors. Its primary features include

- on-chip CPU, FPU, MMU, primary caches, and system interface logic (See Figure 1),¹
- superpipelining techniques,
- on-chip secondary cache control logic with a flexible interface,
- a programmable system interface for high-performance multiprocessor servers and low-cost desktop systems,
- flexible multiprocessor support, and
- 1.2 million transistors implemented in CMOS technology.

In addition, the R4000's single-chip implementation makes it easier to scale the clock as technology improves. According to SPEC benchmark tests, it achieves the highest performance of any microprocessor chip.

A 64-bit architecture

With programs growing by one-half to one bit of address space per year,² a greater than 32-bit address space should be useful by 1993 and required by 1997. In creating the 64-bit R4000, designers extended the R3000 architecture by increasing the data word size and virtual address space. This design entailed widening the machine registers and data paths, and sign-extending 32-bit data when loading into registers. Since certain operations work differently on 64-bit data than on sign-extended 32-bit data, we added additional instructions for 64-bit data, including integer loads, stores, adds, subtracts, shifts, multiplies, divides, and coprocessor moves.

The chip also supports a 64-bit virtual address space with wide virtual address data paths. It stores 32-bit addresses as 64-bit entities in sign-extended form and stores the results of address computation on these entities in sign-extended form. Thus it continues to support the previous 32-bit architecture's addressing.³

The hardware cost of extending the architecture to 64 bits was about 7 percent of the die

area. A longer, 64-bit ALU stage represents the cycle time speed penalty.

CPU pipeline

The R4000's eight pipeline stages allow it to process more instructions at once than can the R3000's five-stage pipeline.⁴ Superpipelining has split the instruction and data memory references across two stages. Consequently, we could distribute the logic more evenly across pipeline stages. (See Figure 2.) The single-cycle ALU stage takes slightly more time than each of the cache access stages.

Although the superpipeline increases the cycles per instruction due to longer branch and load delays, it greatly improves the achievable cycle time. Future increases in cache size will not require a fundamental redesign of the superpipeline. We considered superscalar design as another way to increase instruction-level parallelism, but our studies showed that with current technology the chip could perform higher with a less complex superpipeline.

Figure 3 on the next page shows optimal pipeline movement, completing one instruction every internal clock cycle. The internal, or pipeline, clock rate of the R4000 is twice the external input, or master, clock frequency.

The processor accesses the instruction cache during the instruction first (IF) and instruction second (IS) stages, with a new cache access starting every cycle. The MMU translates the instruction virtual address into a physical address during these stages. The instruction bits available at the beginning of the register file (RF) stage are decoded and used to access the register file. Also at this time, the tags read from the instruction cache are compared with the physical address to determine whether the instruction cache access was a hit. If so, the instruction can advance to its execution (EX) stage. For nonmemory operations, the instruction's result is available by the end of the EX stage.

In the data first (DF) and data second (DS) stages, the R4000 accesses

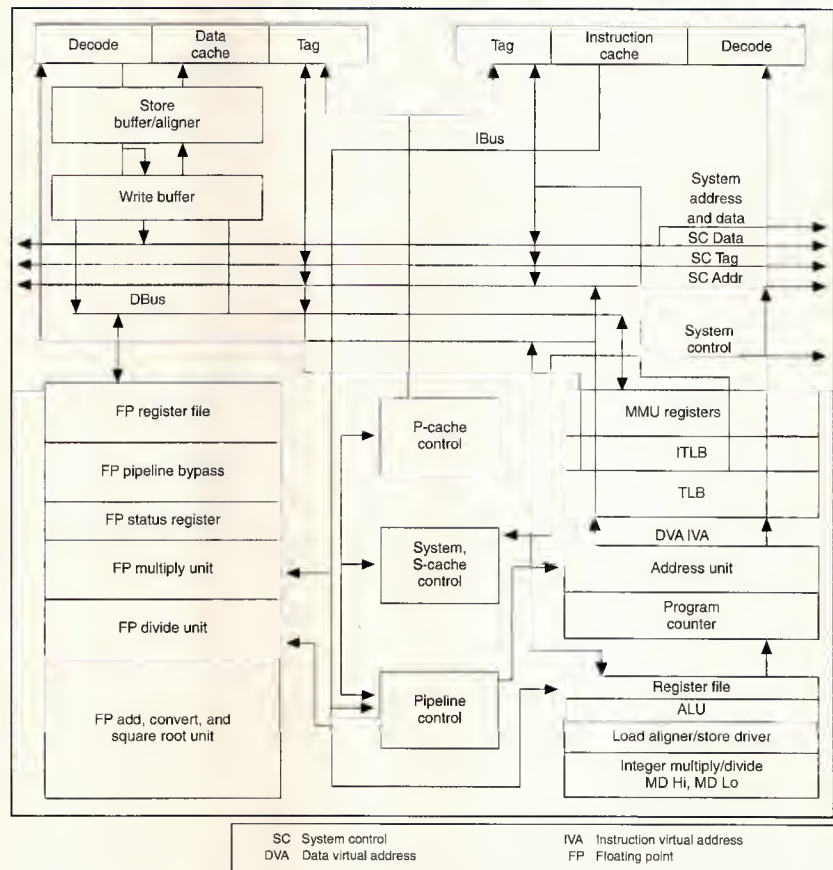


Figure 1. R4000 internal block diagram.

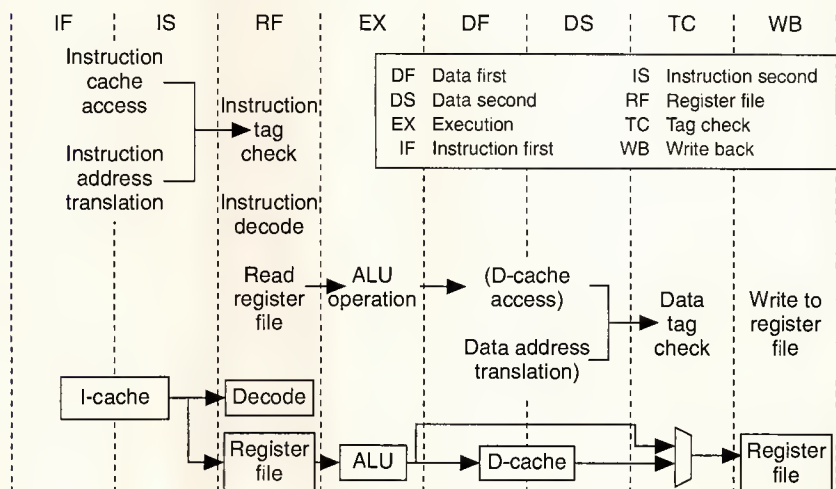


Figure 2. R4000 pipeline activities.

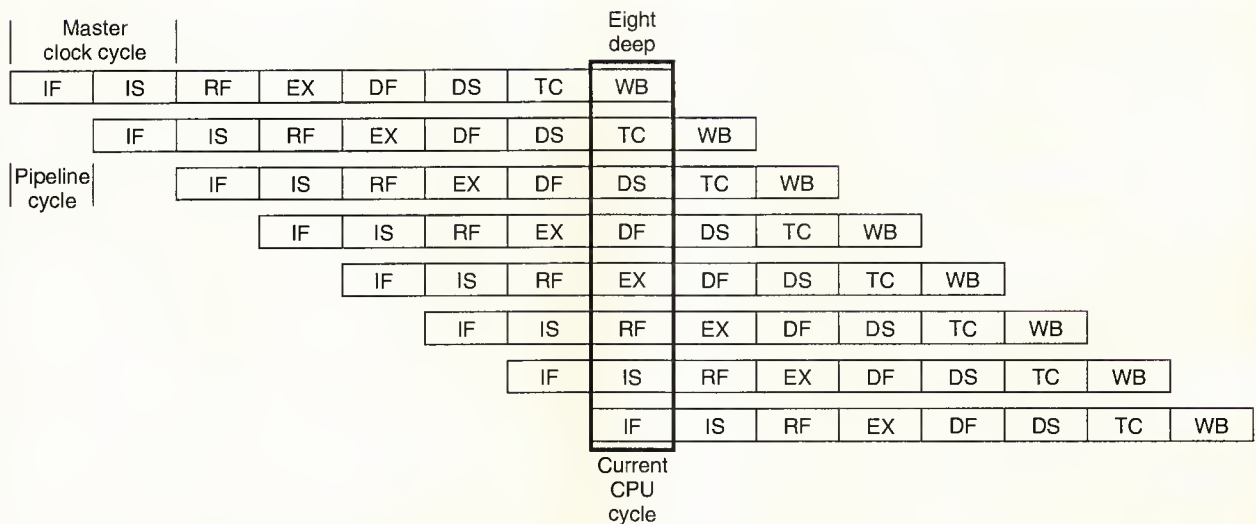


Figure 3. R4000 pipeline and instruction overlapping.

the data cache, with a new access starting every cycle. The MMU translates the data virtual address into a physical address during these stages. In the tag check (TC) stage, the R4000 compares the data tags from the cache tag array with the translated address to determine if the data cache access was a hit. For stores, if the tag check passes in TC, the data travel to the store buffer and enter the data cache the next time cache bandwidth is available. Instructions finally go to the write back (WB) stage where the data are written to the register file if necessary.

Load interlocks and branch instructions disrupt the normal flow of the pipeline. For loads, the data are not ready until the end of the cache access in the DS stage. If any of the two instructions after a load use the result of the load in their EX

stages, the hardware interlocks and slips. As shown in Figure 4, during the slip the DF, DS, TC, WB stages of the pipeline advance while the IF, IS, RF, EX stages do not. For the load interlock, this permits the load instruction to advance and complete its cache access, while the instruction that depends on the load remains in the EX stage.

The result of a branch condition check and a branch target address calculation are not known until the end of the EX stage. (See Figure 5.) By that time, up to three subsequent instructions have entered the pipeline. If the branch is not taken, the processor can continue to execute all instructions that have entered the pipeline with no penalty. If the branch is taken, the processor accesses instructions at the branch target address. For taken branches, the Mips architecture allows one

instruction after the branch to complete before executing the branch target instruction. The other two instructions that have already entered the pipeline are nullified. We considered a branch target scheme that prefetches instructions from both paths of a branch, producing a smaller branch penalty. However, implementation constraints required the simpler approach without a prefetching scheme.

Results of instructions that have completed their execution, but have not yet written their results into the register file, may be bypassed as operands for subsequent instructions.

Integer data path

The R4000's 64-bit execution unit includes a 64-bit register file, load aligner, ALU, shifter, multiplier, and divider. The 64-bit data path supports extended ad-

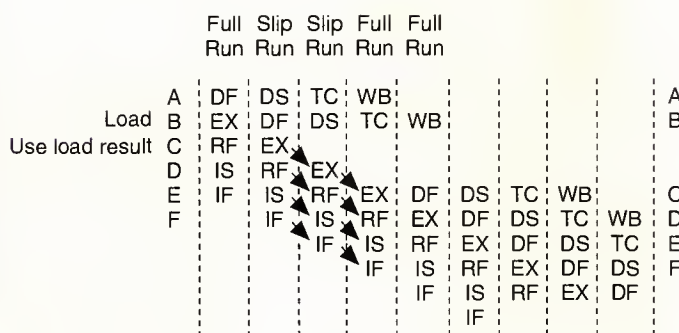


Figure 4. Load interlock/slip cycle.

addressing without the use of long pointers or segment registers.⁵

The ALU stage, EX, was a speed-critical path. To shorten the cycle time, the ALU comprises an adder and a logical unit. The 64-bit, carry-select adder manipulates all 32-bit operands as sign-extended, 64-bit operands. It also performs address calculations for loads, stores, and branches, and is used in integer multiply and divide.

R4000 provides hardware support for integer multiply and divide. It uses a 2-bit Booth algorithm for integer multiplication and breaks each iteration into four stages: Booth decoding, multiplicand selection, partial product generation, and product accumulation. The carry-save adder (CSA) adds intermediate partial products, and two separate 64-bit registers Hi and Lo store the final product.

The multiplier cycles at twice the pipeline clock frequency to produce two sums for each pipeline cycle. Since the R4000 uses a CSA, the multiply results are in a sum-and-carry form and must be combined through full carry propagation. The integer ALU performs this operation when the result moves to the general registers. Integer multiply latency is 10 pipeline cycles for 32-bit operations and 20 pipeline cycles for 64-bit operations.

Divides use a 1-bit-per-iteration, nonrestoring algorithm. This algorithm leaves the quotient in a signed-digit form that must be converted back to a binary representation and possibly corrected at the end of the divide. Divides use the main integer adder for the remainder add or subtract operations, thus preventing the instructions from entering the pipeline during a divide. The implementation takes two pipeline cycles per iteration; each iteration resolves 1 bit of dividend. The latencies are 69 pipeline cycles for a 32-bit divide and 133 pipeline cycles for a 64-bit divide operation. We found this performance sufficient, due to the infrequent occurrence of the integer divide operations.

The integer shifter performs immediate or variable shifts from zero to 63 places. We designed the shifter to shift up to 32 bits in one cycle, making it half the size of a 64-bit shifter. To accomplish shifts greater than 32 bits, the pipeline slips for one cycle while forcing a 32-bit shift in the EX cycle. In the next cycle, the shifter performs the remainder of the shift. A trade-off between area and performance led to this decision.

The register file is a 32-entry by 64-bit array with two read ports and one write port. It can read and write in the same cycle. In the case of reading and writing the same location in the same cycle, the R4000 provides local bypassing of the write data into the read bus.

Floating-point unit

The FPU implements the IEEE Std 754-1985.⁶ Its three functional units—multiplier, adder, and divider—operate on single- and double-precision operands. While the FPU ex-

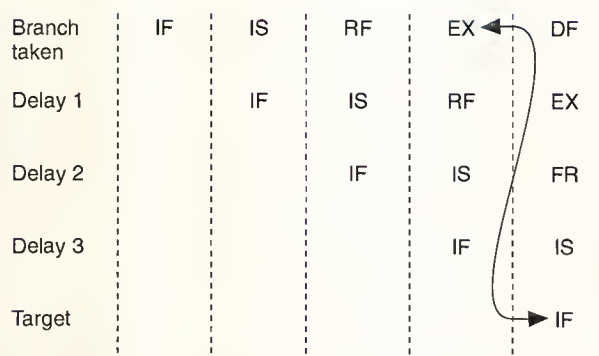


Figure 5. Branch delay.

ecute a multicycle operation, the CPU pipeline can continue in parallel until the FPU detects a data or resource dependency. It can transfer data directly to or from the CPU or cache memory. The FPU executes up to three instructions concurrently, one per functional unit. It retires only one instruction per cycle.⁷

The floating-point multiplier (see Figure 6 on next page) uses a modified Booth algorithm that scans four overlapping groups of 3 bits at once. Thus 8 bits of the multiplier operand can retire with each iteration. The mantissa portion of the multiply array uses four CSAs in a pipeline fashion. The multiplier pipeline includes four stages:

- Booth encoding and multiplicand selection,
- partial sum-and-carry generation of selected multipliers,
- partial product summation of the previous stage result with the previous iteration result, and
- guard, round, and sticky-bit generation.

In the cycle following the last iteration of the multiply, the sum and carry from the multiplier array travel to the floating-point adder to produce the final rounded product.

The multiplier cycles at twice the pipeline clock frequency, so each iteration through the multiplier takes only half a pipeline cycle. R4000's high-speed operation demands that the multiplier array use a two-phase design approach. To reduce the clock skew in this region, the multiplier uses stronger clock drivers (with lower fanout). These drivers allow more aggressive latch designs with improved set-up times, and thus reduce overhead. All CSA and Booth multiplexers use dynamic logic design due to speed criticality.

The floating-point multiply latency is seven pipeline cycles for single-precision and eight for double-precision operations. The repeat rate is three pipeline cycles for single precision and four for double precision.

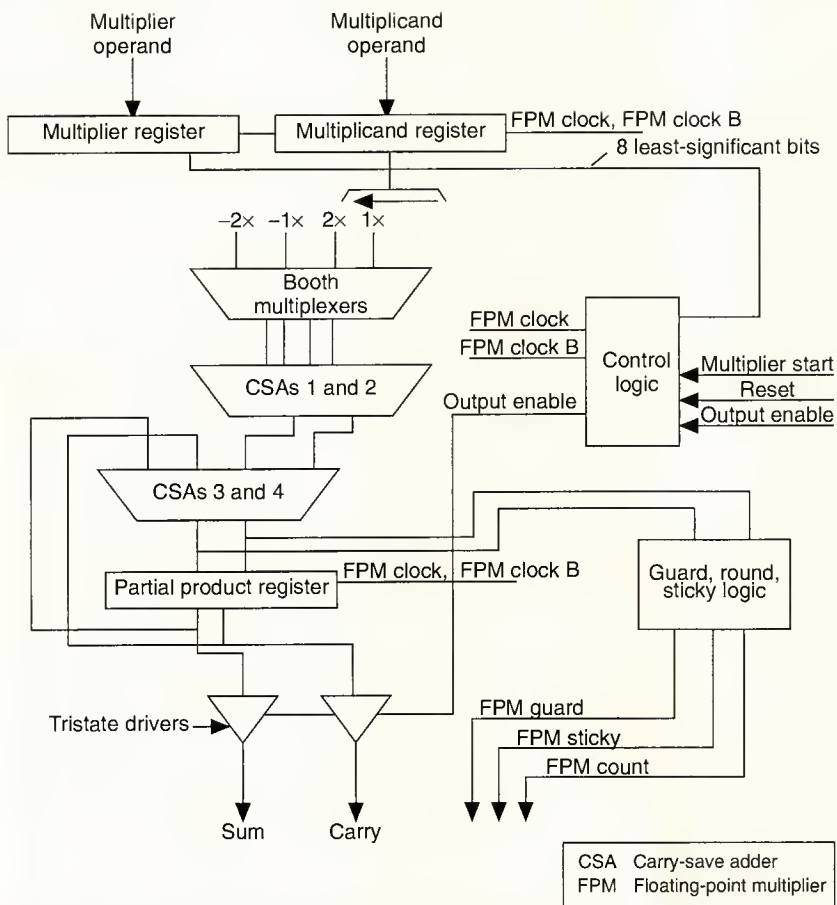


Figure 6. Block diagram of the floating-point multiplier.

The floating-point adder (Figure 7) processes one add or subtract in four pipeline cycles and starts a new operation every three pipeline cycles for both single- and double-precision operations. The adder also assists the multiplier and divider for cleanup operations, such as rounding, and final result computation.

To provide necessary bandwidth to support a two-staged, pipelined multiplier (as seen by the adder), we designed the adder to process a pair of double-precision, multiply-and-add instructions every four cycles.

The adder comprises four stages:

- unpack,
- mantissa add,
- result rounding, and
- mantissa shift (alignment/normalize).

The adder has two data entry paths. One accommodates the

normal source operands that go through the unpack stage to form data inputs for all adder-supported operations. The multiplier/divider units send their intermediate results on the other path to the adder's input stage for final computation. No new instructions can enter the pipeline while the intermediate result travels from multiplier or divider to the adder for the cleanup cycles. The one data repacker in the FPU packs the final result produced by the adder to the correct data format.

We based the floating-point divide operation on the SRT divide algorithm,⁸ which selects the quotient digit based on an estimation of the partial remainder. This technique has the advantage of not requiring a full-precision adder to add or subtract the partial remainder with a divisor multiple. Therefore it runs faster. The latency and repeat rates for floating-point divide operations are 23 and 22 cycles for single-precision operations and 36 and 35 cycles for double-precision operations. (See Table 1.)

The adder calculates square root by generating 1 root bit per cycle using the SRT algorithm. Since the adder also supports multiply and divide instructions, no new computational instruction may start while it calculates a square root. The square-root latency is 54 and 112 cycles for single- and

double-precision operations.

Designers equipped the floating-point divider and the multiplier units with features that allow the circuit to power down at the end of every operation by recirculating zeros in the unit.

The floating-point register file is a 32-entry by 64-bit array with two read ports and two write ports. We dedicated one of the write ports for FP computational result writebacks and the other for FP load, store, and move instructions. In the case of reading and writing the same location in the same cycle, the register file locally bypasses the write data onto the read buses.

Stalls, slips, and exceptions

Pipeline hazards interrupt smooth pipeline flow (Figure 2), causing stalls, slips, or exceptions. In stall cycles, the pipeline does not advance. When the R4000 processes the stall, it restarts the pipeline and reissues several instructions to generate correct results.

For slips, such as the load interlocks detailed earlier, only

the DF, DS, TC, and WB stages advance while the IF, IS, RF, and EX stages do not. When the slip condition is resolved, the instructions in the pipeline resume from whatever stage they are in. For exceptions, the processor suspends the normal sequence of instruction execution and transfers control to an exception handler, detailed later.

Figure 8 on the next page shows how the entire pipeline stalls for a data cache miss on load instruction 1. Since the load miss processing takes several cycles, the pipeline stalls until the secondary cache and main memory access completes. Note that before we got into the stall, instruction 4 may have used erroneous data in its EX stage that was bypassed from the load instruction. During the restart sequence, the processor repeats the EX stage for instruction 4 to obtain the correct data from the LOAD operation. The different stall types include

- *Data cache miss*, detected by the data tag check
- *Data first stage stalls*, which can occur for three mutually exclusive groups of instructions. 1) The pipeline stalls to resolve whether the FP instruction will cause an exception before moving on to guarantee precise exceptions. 2) The pipeline stalls to let the instruction sign extend the result. 3) The pipeline stalls to let the store buffer entries retire to memory because control logic has detected a load to the same memory location.
- *Instruction cache miss*, detected by the instruction tag check
- *Instruction translation look-aside buffer stalls*, for instruction TLB misses (explained in detail later)
- *Multiprocessor*, generated by requests from other processors

Slips occur when the result of an instruction is not available until the DS stage of an instruction, as occurs with loads. Floating-point instructions interlocked for resources also cause slips, as do integer instructions waiting for an integer multiply or divide operation to complete. Variable shifts and shifts greater than 32 bits also use slips since these operations take two cycles to complete.

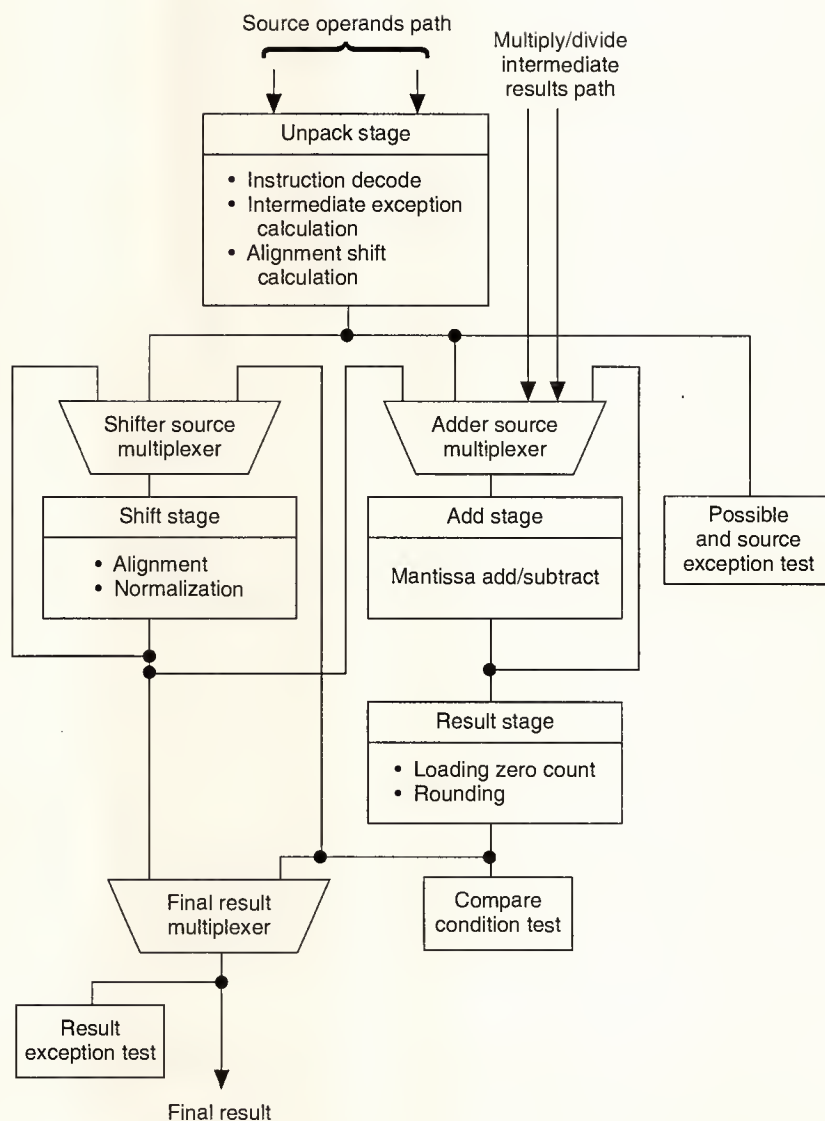


Figure 7. Adder logical block diagram.

Table 1. Integer and floating-point operation latencies and repeat rates in pipeline cycles.

	Integer		Floating point			
	32 bits	64 bits	Latency		Repeat	
			SP	DP	SP	DP
Add/subtract	1	1	4	4	3	3
Multiply	10	20	7	8	3	4
Divide	69	133	23	36	22	35

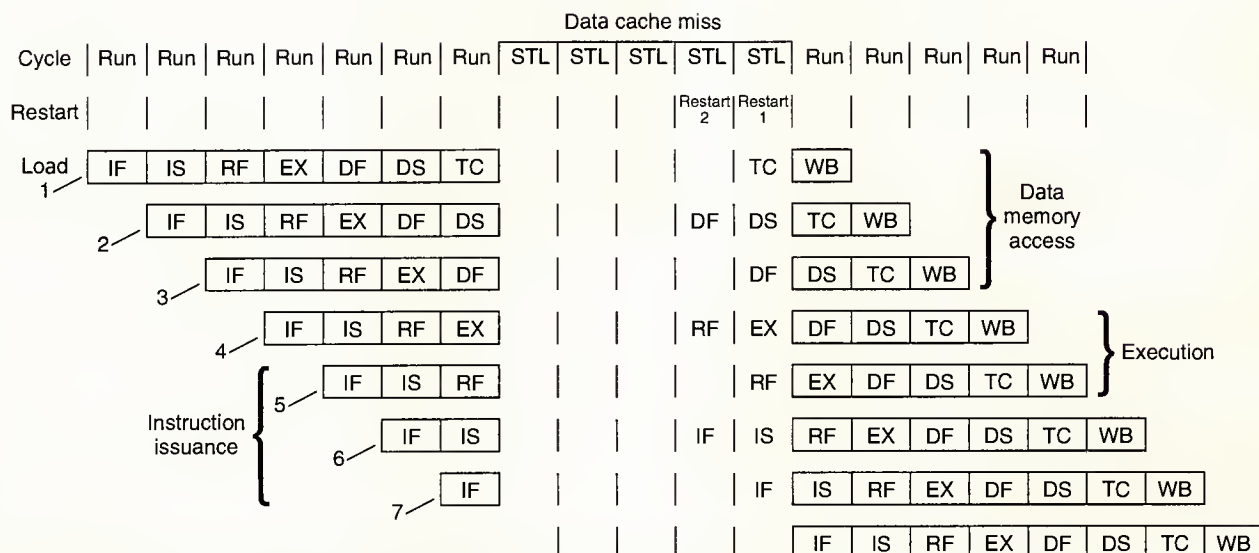


Figure 8. ADD data cache miss, use of load. STL indicates a stall.

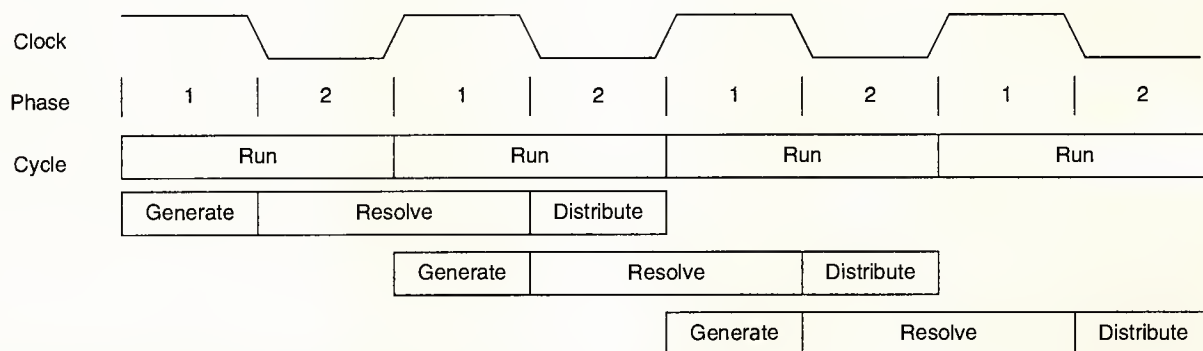


Figure 9. Circuit pipelining.

The R4000 processes many stalls and slips simultaneously. By slipping on instructions that need the same resources as a multicyle floating-point instruction, it can simultaneously accept other stall conditions from instructions that continue to advance further down the pipeline. Also, multiprocessor-initiated stalls, which can stall the pipeline to examine the cache, occur simultaneously with DCT, DFT, and ICT stalls described above.

Stall and slip implementation. The state machines that control pipeline flow (run, slip, and restart machines) operate in a pipelined fashion. When logic detects a stall or slip condition in a given cycle, the soonest the R4000 can process this condition is the end of the next cycle.

Figure 9 shows a sample timing diagram. In the first phase,

the pipeline control unit evaluates logic that may generate a stall or slip condition. In phase 2 and the second phase 1, the state machines are resolved. Finally, the pipeline control signals are distributed throughout the chip during the second phase 2.

After processing a stall, the R4000 initiates a two-cycle restart sequence before the pipeline can run again. During this sequence, it reevaluates portions of the pipeline with corrected information before normal pipeline flow resumes. As shown in Figure 8, it repeats three activities: data memory access, execution, and instruction issuance.

Exception handling. The R4000 processes exceptions from sources in different pipeline stages. It prioritizes incoming exceptions and gives highest priority to the faulting instruction furthest along the pipeline. Table 2 lists different

exceptions and the stages where they are signaled.

During normal processing, the R4000 nullifies pipeline stages for three reasons.

- When an exception occurs, it nullifies instructions after the faulting instruction.
- It nullifies certain instructions in branch delay slots when a branch is taken.
- When the pipeline slips, it creates a nullified instruction "bubble," as the back end of the pipeline advances and the front end does not.

After being nullified, the instruction does not commit to any state. For performance, the processor inhibits any stalls signalled by the instruction. For example, if an instruction will cause a data translation exception, which is detected at the end of the DS stage, the processor will not allow it to signal a cache miss in the TC stage.

Memory management unit

The MMU translates virtual addresses into physical addresses using an on-chip translation look-aside buffer (TLB). It manages exceptions, controls the cache subsystem, and provides diagnostic and error recovery facilities. Compared to the R3000, the R4000 MMU provides enhanced operating system support including increased TLB entries, variable page sizes, 64-bit architecture support, supervisor privilege level, timer interrupts, and a physical address trap.

We wanted to increase the number of entries in the TLB over the 64 entries available in the R3000 since this boosts performance in a wide range of applications. Using 128 entries required too much area for the fully associative lookup circuit. Therefore, we implemented a 48-entry TLB with each entry mapping two consecutive pages and producing 96 effective entries. The TLB superpipelines in the R4000 (across the DF/DS pipeline stages) and runs in parallel with the cache access.

The instruction translation look-aside buffer (ITLB) is a two-entry, fully associative translation buffer that is a subset of the main TLB. This ITLB supports only a 4-Kbyte page size, to reduce complexity with minimum performance impact. When an instruction miss occurs in the instruction buffer, the pipeline stalls and the main TLB refills the ITLB. When a branch is taken into a different page, the branch target instruction address translation uses the TLB bandwidth available during the data first and data second stages of the branch instruction. Since the instruction first and instruction second stages of the branch target line up with the data first and data second stages of the branch instruction, the target address translation refills the ITLB without stalling the pipeline.

The R4000 implements variable page sizes on a per-page basis, varying from 4 Kbytes to 16 Mbytes. This helps to reduce thrashing of the TLB in some cases, such as in the use of a frame buffer which uses large data blocks. It implements

Table 2. Exceptions.

Cycles	Exceptions
IF	—
IS	—
RF	Instruction translation
EX	Interrupt
	Bus error instruction
	Illegal instruction
	Breakpoint
	Syscall
	Coprocessor unusable
	ECC instruction
	Virtual coherency instruction
DF	—
DS	Overflow
	Floating point
TC	TLB modified
	Data translation
WB	Bus error data
	Virtual coherency data
	Watch
	NMI
	Reset

variable page sizes by having a mask associated with each TLB entry. When addresses approach the TLB for translation, the corresponding mask bits in the TLB specify which virtual address bits participate in the comparison and translation.

The R4000 instruction set architecture supports 64-bit addressing. The current revision of the R4000 uses 40 bits of the 64-bit virtual address space. Increasing the effective virtual address size above 40 bits would have made the TLB wider than the data path and difficult to fit into the layout. Hardware explicitly checks the unused upper bits (bits 61:40) of the virtual address to make sure they are zero, ensuring a smooth transition for software as the size of the virtual address grows in future revisions. The R4000 supports a physical address of 36 bits.

The unit includes a supervisor privilege level of operation, in addition to the kernel and user levels present in previous company designs. This mode improves operating system support with more privilege levels.

A CACHE instruction provides a set of operations allowing the implementation of both a high-performance, symmetric, multiprocessing operating system and a high-performance workstation operating system. This instruction makes some tasks more efficient, including block copy, page zeroing, cache initialization, page flushing, and cache testing.

The CACHE instruction supports a number of operations including

- load and store of cache tags,
- selective invalidation of cache lines,
- create dirty exclusive data cache lines, and
- forced writeback of lines.

The R4000 provides a physical address trap feature for debugging software. This takes an exception on a reference to a selected physical address, which is specified in the Watch register.

The Count and Compare registers implement a timer interrupt service. The Count register acts as a timer, incrementing at half the pipeline clock rate. When the value in the Count register equals the value in the Compare register an interrupt occurs.

Memory hierarchy

The R4000 fits a range of system configurations. A programmable system interface permits tuning to different system specifications and exploiting future improvements in DRAM and SRAM design. The R4000 supports a two-level cache hierarchy that configures to run with different line sizes. Multiple cache coherency protocols available on the R4000 support several multiprocessor systems.^{9,10}

The limited available primary cache size necessitated support for a closely coupled off-chip secondary cache required by high-end systems. We estimated the cache control section required 10 percent extra logic to support systems both with and without secondary cache. The R4000 manages its primary and secondary caches using a write-back method, in which stores send data into the caches, but the data do not write back to memory until the cache line is replaced or flushed.

The processor maintains its primary caches as a subset of the secondary cache contents. This prevents the occurrence of virtual aliases, which could lead to incorrect operation. A virtual alias occurs when multiple virtual addresses in the primary cache map to the same physical address in the secondary cache.

The primary caches are virtually indexed, so the secondary cache stores 3 bits of the virtual address (bits 14 to 12) needed to locate the primary cache lines that may contain data from a particular secondary cache line. (This virtual address information will support primary caches up to 32 Kbytes each). Because only one copy of the secondary cache line can reside in the primary cache, no two virtual addresses in the primary cache can map to the same physical location. Without this capability, R4000 would have to flush the large secondary cache to prevent aliasing. This is time consuming, especially for aliases caused by reusing pages for I/O.

Primary cache. While the initial version of R4000 uses an on-chip primary cache size of 8 Kbytes of instruction and 8 Kbytes of data, we can easily increase these sizes. The current revision supports primary caches up to 32 Kbytes each of instruction and data.

The primary cache is a direct-mapped, virtually indexed, physically tagged cache. Direct mapping makes it easy to find the location of a particular line in the cache and to manage

cache consistency between the primary and secondary caches.

As the primary cache is virtually indexed, the virtual address generated by R4000's address unit looks up the cache line, while the address translation occurs in parallel. The address translation produces the physical address of the access, and the comparator compares it with the physical address read from the tag of the cache lines. The processor uses data coming out of the cache before it checks the tag, reducing the delay before load data can be used by one cycle.

Direct-mapped caches access faster than associative caches, but their hit rate is not as high as for set-associative caches. This penalty decreases as we increase the size of the primary caches. The primary caches support two software-programmable line sizes (16 and 32 bytes) that users can change independently for the instruction and data caches.

R4000 needs two cycles to access data in the primary cache, but a new address may enter every cycle. This is possible because the processor accesses the cache array in one cycle, excluding the address buffering and the data drive time. The address does not access the array until the beginning of phase 2 of the first cycle, when the data from the previous access have been latched.

The primary instruction and data caches have separate data and tag arrays. The data cache data array and tag array may be addressed separately every cycle. During the data first and data second stages of a store instruction, the processor accesses the tag array for the store, while it may access the data array for a previous store that has passed its tag check and has data waiting in the store buffer. The two-entry store buffer decouples the data to be stored from the rest of the pipeline.

Since the architecture supports byte stores, the data cache array is arranged in eight blocks. Each block has a byte of data, a parity bit, and a redundant bit. The primary caches access 64 bits of data at a time, with the ability to write selected bytes. Row and column redundancy terms improve the die yield. To replace a defective row or column in one of the cache arrays with a redundant row or column, the manufacturer must blow the laser fuses.

Secondary cache. The secondary cache is direct mapped, physically indexed, and physically tagged. Manufacturers can build it from industry-standard static RAMs of different speeds and densities. The 128-bit-wide secondary cache interface allows a single access to the secondary cache to fill a four-word primary cache line. This cache supports a line size of four, eight, 16, or 32 words.

A physically indexed secondary cache makes multiprocessor support easy as all addresses on the system bus can be physical, eliminating the need for extra address translation information.

With R4000 supporting a maximum secondary cache size of 4 Mbytes, and with several such caches present in a multiprocessor system, the probability of a soft error demands support for error checking and correction. This ECC support for the secondary cache corrects 1-bit errors and detects 2-bit

errors. R4000 performs on-chip tag correction, but it needs external hardware support to correct data errors.

We chose parity support for the primary cache since the on-chip caches are small and less prone to soft-error failure. If the operating system finds a parity error in the primary cache on a clean line, it can arrange to refill the primary cache line. When it detects a cache error, the processor takes an exception and jumps to uncached space. There the operating system examines the cache error control register, which specifies the type and location of the cache error.

One complex operation carried out in the cache logic is the write-back of dirty lines to memory. During writebacks, a state machine, the *zipper*, merges dirty (corrupted) lines in the primary cache with the data from the secondary cache as the line transfers to the system interface. The zipper checks tags in the primary instruction and data caches. It invalidates both instruction and data lines while merging any dirty data from the primary data cache. This operation completes in four pipeline cycles to match the maximum speed supported by the secondary cache.

System interface. The system interface lets the processor access external resources required to satisfy cache misses. It also allows an external agent access to some of the processor's internal resources. For multiprocessor systems, the system interface provides the processor mechanisms necessary to maintain cache coherence of shared data.

R4000 uses a 64-bit-wide system interface to increase main memory bandwidth compared with previous 32-bit system interfaces. The system interface can receive a double word every two pipeline cycles. If R4000 is operating without a secondary cache, the system interface can operate at the maximum system interface data rate, since the primary cache has a 64-bit data path that supports this rate. With a secondary cache, the maximum data rate the processor can support directly relates to the secondary cache access time. If the access takes too long, the processor cannot transmit or accept data at the maximum rate. The secondary cache only accepts reads and writes occurring in at least four cycles. With fast static RAMs that support a four-cycle access, the secondary cache interface can keep up with data coming in from the system interface at the maximum rate. Designers can program the system interface to transmit data in a range of rates, to suit different system and secondary cache speeds.

The system interface can be programmed to be clocked by a divided-down version (divided by two, three, or four) of the internal clock frequency. The internal clock runs at twice the processor's input, or master, clock. This allows systems designed for slower ver-

sions of the R4000 to run faster versions. For example, a system designed for a 50 MHz R4000 (with the system interface programmed to halve the internal 100 MHz pipeline clock) could implement a 75 MHz R4000 with the system interface clock divisor changed to divide by three. A 75 MHz external clock generates a 150 MHz internal pipeline clock, which the divisor divides by three to produce a 50 MHz system clock.

The R4000 supports an overlapped mode of operation on the system interface when configured with a secondary cache. When a miss occurs in the secondary cache that requires a line to be written back to main memory, the system interface sends out a read request for the miss and then immediately sends out a write with the writeback data. This saves the R4000 from having to buffer up secondary cache lines before they are written back, which would use significant chip area to support the largest secondary cache line of 32 words.

Multiprocessor support. The R4000 provides mechanisms to implement a variety of cache coherence protocols that may be snoopy or directory based (see Figure 10). Designers closely coupled the multiprocessor logic with the pipeline activity to allow access to the primary caches.

The starting point for R4000's coherence model was the MESI (modified, exclusive, shared, invalid) protocol. MESI implements a four-state cache coherence protocol (the states are invalid, clean exclusive, dirty exclusive, and shared). R4000 implements a fifth, the dirty shared state (Figure 11 on the next page), which allows for efficient implementations of a semaphore given the support for update protocol. When a processor successfully acquires a semaphore by gaining a dirty shared copy of the semaphore, all the other processors using that semaphore will be updated with its new value. They don't need to generate additional transactions on the bus. With the MESI protocol, a request from another processor (that is, an intervention) can cause writebacks to the sys-

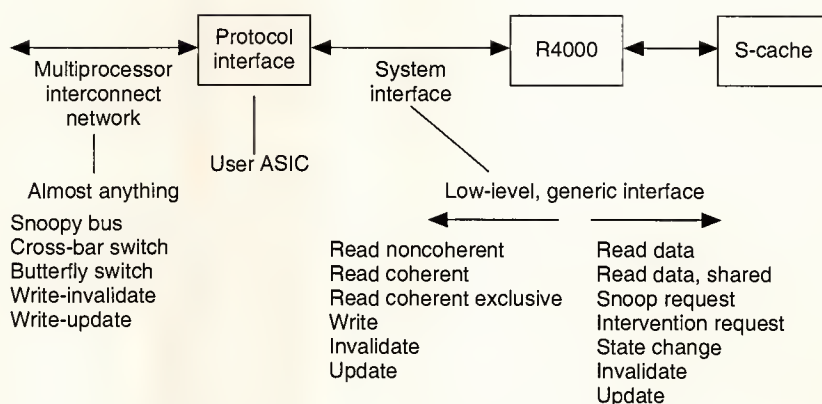


Figure 10. Multiprocessor protocols.

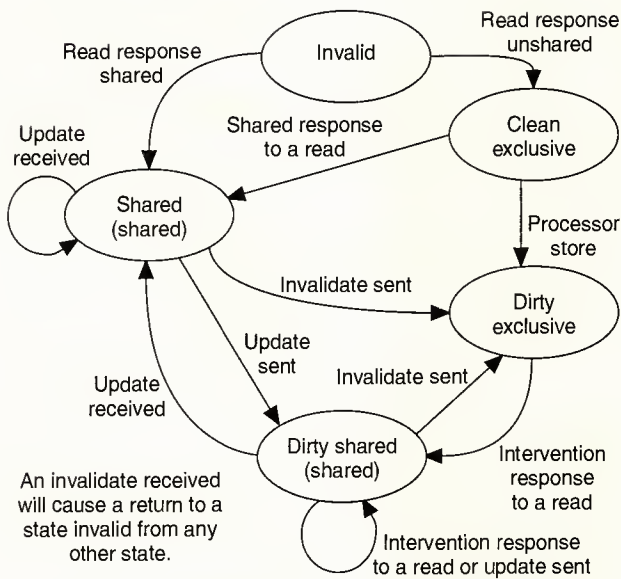


Figure 11. Cache coherency diagram.

tem memory. These writebacks place an additional burden on the system design. (The R4000 cannot process three-party transactions on interventions.) The processor stores the state of a cache line along with the tag and data for each line in the caches.

When R4000 receives an external snoop, intervention, invalidate, or update, it checks the secondary cache tag and state bits while allowing the processor to operate within the primary cache space in parallel. Misses in the secondary cache require no further action because the primary is a subset of the secondary. If an external event hits in the secondary cache, access to the primary may be required to complete the transaction. To gain access to the primary cache, the processor stalls the CPU pipeline.

The processor supports write-invalidate and write-update protocols, controlled on a per-page basis. The TLB may mark pages as uncached, noncoherent, coherent exclusive, coherent-write exclusive, and coherent-write update. Table 3 shows examples of the actions caused by these attributes.

The R4000 provides a load linked and store conditional pair of instructions to provide synchronization between processors on the system bus based only on cache coherency. An example of this is the fetch-and-add operation.

```

Loop: ll      T0,0 (T1)    ;load counter, set load link bit
      addu   T0, T0, 1    ;increment
      sc     T0, 0 (T1)   ;store back if load link bit still set
      beq    T0, 0, Loop  ;retry if store failed

```

The store conditional instruction fails if the location has been invalidated or updated since the preceding load linked instruction. This mechanism can implement semaphores, bitlocks, fetch-and-add, and other synchronization mechanisms. It also guarantees that at least one processor on the bus will get the semaphore on the first attempt so deadlocks or long stalls will not occur.

Design methodology

We chose full-custom data path layout for maximum speed and the highest packing density. Designers implemented most of the control sections using a logic synthesis and optimization tool and laid them out using standard cell place-and-route methodology. However, to achieve our target cycle times, we had to custom design and lay out by hand some of the control sections in the critical paths.

We used a two-phase, zero-overlap clock strategy and distributed it throughout the chip with a balanced clock tree, to control skew. A phase-locked loop generates four times the frequency of the external input (master) clock and distributes it through the chip. Divide-by modules at the end of the clock tree generate 2 \times - and 1 \times clocks. The processor pipeline and most logic use the 2 \times clock, which cycles twice as fast as the master clock frequency. The integer multiplier and floating-point multiplier use the high-speed 4 \times clock, four times the master clock frequency.

The chip uses two types of register/latches: stacked and pass-gate dynamic. Stacked registers, used extensively, are immune to clock skew as long as there are zero or an even number of inversions between the two stacked latches. However, when a short setup time and fast clock-to-output delay

were necessary, we used pass-gate dynamic latches. In these cases, a design rule enforced a delay equivalent to the time needed to pass through at least three inverters of a fan-out of three between the latches to prevent data slip-through.

We equipped the output buffers with a digitally controlled slew rate to reduce noise injected into the system buses. One buffer determines the digital control signal values for the rest of the buffers. This output buffer sends the pad a signal, which in turn feeds

Table 3. Examples of actions caused by coherency attributes.

Algorithm	Load-miss	Store-miss
Uncached	Word read	Word write
Noncoherent	Block read noncoherent	Block read noncoherent
Coherent exclusive	Block read exclusive	Block read exclusive
Coherent write exclusive	Block read	Block read exclusive
Coherent write update	Block read	Block read/update

back into an input pad. The processor samples the round-trip delay and references it with the clock cycle. Users can program the desired amount of skew in terms of a fraction of a clock cycle. Depending on the control signals generated, the strength of the output buffer's pullup and pull downs are adjusted. (See Figure 12.)

We laid out the chip using a generic Mips design rule, so all our semiconductor partners can work from a single database. This database is based on a 1.0- μ m-drawn, two-layer metal, CMOS technology. Manufacturers are producing the R4000 in 0.8 μ technology.

Verification

Mips carried out a functional simulation of a register transfer level model during the development of the R4000. The RTL model executes at about 1,000 processor cycles per minute on a 20-MIPS, R3000-based Magnum workstation. Designers divided the chip into major functional blocks (CPU, FPU, MMU, caches, and system interface) and wrote directed diagnostic tests to exercise these functional units. Trace comparisons of diagnostic tests run on an instruction-level simulator and on the R4000 RTL verified compliance with our architecture. To trace all the required signals and data in the R4000 superpipeline, we added more verification logic to the R4000 RTL model so it could capture traces for comparison with the instruction-level simulator traces.

We performed extensive automatically generated random diagnostic tests, again using our instruction-level simulator for trace comparison. We wrote additional verification diagnostics to ensure that all the arcs of the state machines within the R4000 were exercised. Our designers executed R4000 diagnostics within an RTL model of a system configurable at runtime to include a secondary cache and change any of the programmable parameters that control the system interface. They booted the Unix operating system on the R4000 RTL model about six months before Mips gave the design to its manufacturing partners. It took a 50-MIPS Mips 6280 seven days of processing to reach the Unix prompt.

We verified the multiprocessor capabilities of the R4000 using a number of different simulation models. A uniprocessor RTL simulation of the R4000 checked that the R4000 could generate and process all the multiprocessor requests defined by the R4000 interface specification. We also developed a simulation environment that could support multiple R4000 processors at the RTL level. Under this environment we ran directed diagnostic tests and self-checking random tests.


Finally, we verified that the physical

implementation of the R4000 matched the RTL description by generating a gate-level model from schematics. Obviously, this model ran much slower than the RTL model, and so we needed a large compute resource to run the diagnostic test suite at the gate level. In the final stages of verification we used ten 6280 machines and around thirty 20-MIPS Magnum workstations.

Testability and packaging

The R4000 implements JTAG (IEEE Std. 1149.1) boundary scan specifications, intended to provide a test capability for the interconnection between the R4000 processor, the printed circuit board, and other components on the board.

The chip comes in two package configurations. The R4000MC and R4000SC, which have the 128-bit data interface to the secondary cache, are packaged in a 447-pin lead or plastic grid array. The R4000MC supports multiprocessor systems while the R4000SC supports high-performance uniprocessor systems. The R4000PC, for desktop, low-end servers, and embedded control systems, comes in a 179-pin PGA with no secondary cache interface.

TABLE 4 LISTS SPECMARKS FOR SIMULATED RESULTS of a realistic memory system. (See next page). We simulated the CPU time and most of the important aspects of memory and heuristically added the I/O times. Correlation of simulations with R4000 systems in the lab show the simulations to be pessimistic. 

Acknowledgments

The R4000 became a reality due to an enormous team effort managed by our leader, Tom Riordan. We also had

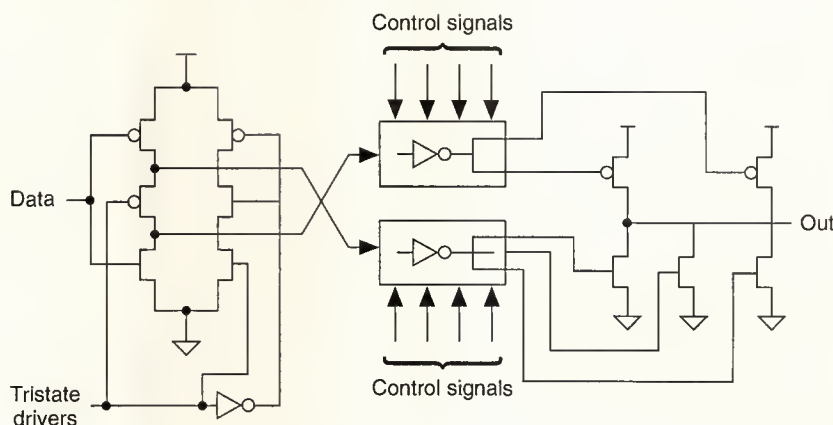


Figure 12. Output buffer.

Table 4. Simulated Specmarks for a 50-MHz external-clock R4000.

Benchmark	S-cache size		P-cache only
	4 Mbytes	512 Kbytes	
Gcc	46	43	27
Espresso	54	54	38
Spice2g6	42	38	27
Doduc	49	46	33
Nasa7	56	46	43
Li	66	65	47
Eqntott	54	52	50
Matrix300	278	273	177
Fpppp	55	54	29
Tomcatv	58	59	37
Simulated SPEC	63	59	42
Simulated SPEC int	55	53	39
Simulated SPEC fp	69	64	44
CPI (simulated SPEC)	1.5	1.6	2.3

great support from other groups within Mips that had to put up with our constant demands for support. Unfortunately, we cannot list each member of the R4000 team, but we thank them all.

References

1. J. Hennessy et al., "Mips: A VLSI Processor Architecture," Tech. Report 223, Computer Systems Laboratory, Stanford University, Stanford, Calif., 1983.
2. J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, San Mateo, Calif., 1990.
3. J.R. Mashey, "64-Bit Computing," *Byte*, Sept. 1991, pp. 135-142.
4. *MIPS R4000 Microprocessor User's Manual*, Mips Computer Systems Inc., Sunnyvale, Calif., 1991.
5. S. Waser and M. Flynn, *Introduction to Arithmetic for Digital Systems Designers*, Holt, Rinehart, and Winston, New York, 1982.
6. *ANSI/IEEE Std. 745-1985, Standard for Binary Floating-point Arithmetic*, IEEE, New York, 1985.
7. C. Rowen, M. Johnson, and P. Ries, "The Mips R3010 Floating-Point Coprocessor," *IEEE Micro*, Vol. 8, No. 3, June 1988, pp. 53-62.
8. D.E. Atkins, "High-Radix Division Using Estimates of the Divisor and Partial Reminders," *IEEE Trans. Computers*, Vol. C-17, No. 10, Oct. 1968, pp. 925-934.
9. S.A. Przybylski, *Cache and Memory Hierarchy Design*, Morgan Kaufmann, 1990.
10. A.J. Smith, "Cache Memories," *Computing Surveys*, Vol 14, No. 3, Sept. 1982.



Sunil S. Mirapuri defines and designs advanced microprocessor products at Mips Computer Systems. Previously, he worked for Rolm Milspec Computers and Intel Japan. His research interests include computer architecture, digital systems, and computer programming.

Mirapuri received his BS and MS degrees in electrical engineering from Stanford University. He is a member of the IEEE Computer Society.



Michael S. Woodacre is a member of Mips' VLSI design verification team. Prior to joining Mips to work on the R4000, he was a VLSI design engineer for Inmos' transputer microprocessor team.

Woodacre received a BS in computer systems engineering from the University of Kent in Canterbury, England.



Nader Vasseghi is a member of Mips' VLSI design group. He worked on the design and development of the R4000 and now works on the next-generation RISC processor. Prior to joining Mips, he designed network controller products and high-speed programmable array logic devices at Advanced Micro Devices.

Vasseghi received his BSEE from the University of California at Santa Barbara and his MSEE from Southampton University in England. He is a member of the IEEE Computer Society.

Address questions regarding this article to Sunil Mirapuri at Mips Computer Systems, 928 Arques Ave., Sunnyvale, CA 94086; or via e-mail at sunil@mips.com.

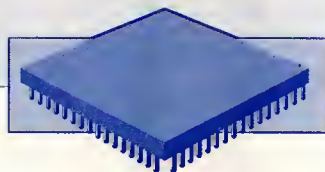
Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 153

Medium 154

High 155



The Message-Driven Processor:

A Multicomputer Processing Node with Efficient Mechanisms

The Message-Driven Processor, an integrated multicomputer node, provides efficient mechanisms for parallel computing. Rather than being specialized for a single model of computation, the MDP incorporates primitive mechanisms for communication, synchronization, and naming. These mechanisms efficiently support most proposed parallel programming models. Each processing node of MIT's J-Machine consists of an MDP with 1 Mbyte of DRAM. MDPs have been operational since June 1991, and J-Machines built from them went on line in July 1991.

William J. Dally

J.A. Stuart Fiske

John S. Keen

Richard A. Lethin

Michael D. Noakes

Peter R. Nuth

*Massachusetts Institute
of Technology*

Roy E. Davison

*Davison Design and
Development Corp.*

Gregory A. Fyler

Intel Corporation

The Message-Driven Processor is a 36-bit, 1.1-million transistor, VLSI microcomputer specialized to operate efficiently in a multicomputer. The MDP chip includes a processor, a 4,096-word by 36-bit memory, and a network port. An on-chip memory controller with error checking and correction (ECC) permits local memory to be expanded to one million words by adding external DRAM chips.

The processor is message-driven in the sense that it processes in response to messages, via the dispatch mechanism. No receive instruction is needed. The MDP creates a task to handle each arriving message. Messages carrying these tasks advance, or drive, each computation.

We designed the MDP with two primary goals in mind.

- We wanted to implement a general-purpose, multicomputer processing node that provides the communication, synchronization, and naming mechanisms required to efficiently support several different parallel programming models.
- We wanted to create an inexpensive, VLSI component for cost-efficient parallel computers. Ideal nodes should be inexpensive and plentiful VLSI commodity parts—as inexpensive and plentiful as jellybean can-

dies—that can network together to form a Jellybean Machine (J-Machine) multicomputer.

Efficient parallel mechanisms

Computer hardware provides primitive operations called mechanisms. These mechanisms build the abstractions that in turn make up a programming system.¹ For example, most sequential machines provide some mechanism for a push-down stack to support the last-in-first-out (LIFO) storage allocation required by many sequential programming models. Most machines also provide some form of memory relocation and protection to allow several processes to coexist in memory at once without interference. The proper set of mechanisms can significantly improve performance over a brute-force interpretation of a programming model.

Over the past 40 years, sequential von Neumann processors have evolved a set of mechanisms appropriate for supporting most sequential programming models. It is clear, however, from efforts to build concurrent machines by wiring together many sequential processors, that these highly evolved sequential mechanisms do not adequately support most parallel models of computation. These mechanisms do not efficiently support synchronization of events, communication of data, or global naming of objects. As a

result, designers must implement these functions, inherent to any parallel model of computation, largely in software with prohibitive overhead. For example, sequential machines require hundreds of instructions to create a new process. This cost prohibits the use of fine-grain programming models where processes typically last only a few tens of instructions.

The MDP supports a broad range of parallel programming models, including shared-memory,² data parallel,³ dataflow,⁴ actors,⁵ and explicit message-passing,⁶ by providing low-overhead primitive mechanisms for communication, synchronization, and naming. Its communication mechanisms permit a user-level task on one node to send a message to any other node in a 4,096-node machine in less than 2 μ s. This process doesn't consume any processing resources on intermediate nodes, and it automatically allocates buffer memory on the receiving node. On message arrival, the receiving node creates and dispatches a task in less than 1 μ s.

Presence tags provide synchronization on all storage locations. Three separate register sets allow fast task switching. A translation mechanism maintains bindings between arbitrary names and values, and supports a global virtual address space. We selected these mechanisms to be both general and amenable to efficient hardware implementation. To support fine-grain, concurrent programming systems, we designed the mechanisms to efficiently handle small objects (eight words) and small tasks (20 instructions).

3D array of fine-grain, processing nodes

The MDP is an example of an inexpensive, fine-grain, multicomputer building block. A fine-grain node does not necessarily have a slow processor. We can build a competent processor in a fraction of a modern VLSI chip's area. Fine grain and small memory decrease the chip's cost, resulting in greater arithmetic performance and local memory bandwidth per unit cost. Fast communication and a global address space prevent the small local memories from limiting programmability or performance.

In a multicomputer, system cost is very sensitive to processor cost. A less-expensive node results in a comparably priced system with more processors and, to first order, higher performance. In these systems, designers avoid costly features that give a small incremental return in processor performance (such as large caches) in favor of building systems with more nodes, an option not available to the designer of a sequential computer.

The 3D network that connects MDPs gives the highest throughput and lowest latency for a given wire density.⁷ This network allows the processing nodes to be packed densely and results in uniformly short wires. It does not waste communication bandwidth by embedding an esoteric topology into physical space. Messages traveling through the network follow a Manhattan shortest path in physical space; they never backtrack. (A Manhattan path travels forward, to the side,

and up or down, but not across diagonals.)

Background

The MDP builds on previous work in multicomputer design. Like the Caltech Cosmic Cube,⁶ Intel's iPSC,⁸ the Ncube,⁹ and the Ametek,¹⁰ each MDP in the J-Machine has a local memory and communicates with other nodes by passing messages. Because of its low overhead, the MDP can exploit concurrency at a much finer grain than these early message-passing multicomputers. Delivering a message and dispatching a task in response to the message's arrival takes less than 2 μ s on the J-Machine, as opposed to 5 ms on an iPSC-1 or 300 μ s on an iPSC-2.

Like the BBN Butterfly¹¹ and the IBM RP3,¹² the MDP supports a global virtual address space. The same IDs (virtual addresses) reference local (on the same node) and remote (on a different node) objects. Like the Inmos transputer,¹³ the Caltech Mosaic,¹⁴ and the Intel iWarp,¹⁵ the MDP is a single-chip processing element integrating a processor, memory, and a communication unit. The MDP is unique because it extends these previous efforts with efficient primitive mechanisms for communication, synchronization, and naming.¹ It uses a direct communication network based on work reported by Dally,⁷ Dally and Seitz,¹⁶ and Dally and Song.¹⁷

System architecture

To the hardware designer, the MDP appears as a component with a memory port, six two-way network ports, and a diagnostic port, as shown in Figure 1.

The memory port provides a direct (that is, no glue) interface to up to 1 Mwords of ECC DRAM, consisting of 11 multiplexed address lines, a 12-bit data bus, and three control signals. Static-column or page mode DRAMs cycle three times to access a 36-bit data word and a fourth time to check or update the ECC check bits. Current J-Machines use three 1M \times 4 memory parts to form a four-chip processing node with

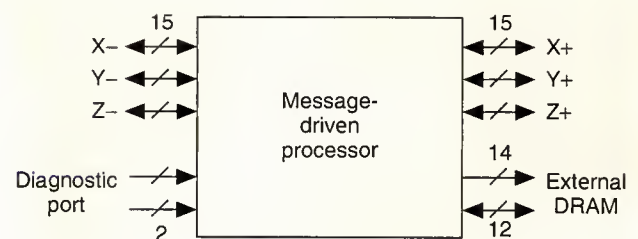


Figure 1. MDP pinout. The MDP has a memory port (26 pins), six network ports (15 pins each), and a diagnostic port (three pins).

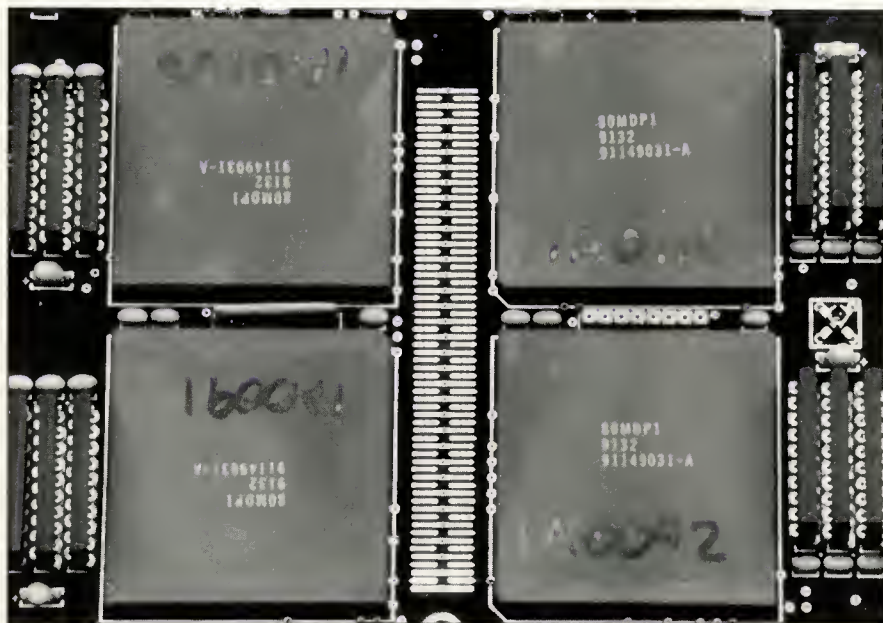


Figure 2. An array of four J-Machine processing nodes. Each node consists of one MDP chip and three $1\text{M} \times 4$ static-column DRAMs. With conventional packaging the node measures 2 in. \times 2.75 in.

262,144 words of memory that measures 2 in. \times 2.75 in., as shown in Figure 2.

The network ports connect MDPs together in a 3D mesh network. Each of the six network ports corresponds to one of the six cardinal directions ($+X, -X, +Y, -Y, +Z, -Z$) and consists of nine data and six control lines. Each port connects directly to the opposite port on an adjacent MDP. We give details of the 3D network later in this article.

The diagnostic port issues supervisory commands and reads and writes MDP memory from a console processor. The port consists of two control lines, a serial input line, and a serial output line. Using this port, a console processor can read or write any location in the MDP's address space, as well as reset, interrupt, halt, or single-step the processor.

Software. To a systems programmer, a bare J-Machine appears as a collection of node memories and register files operable by an instruction set that includes communication, synchronization, and naming mechanisms. The systems programmer uses these mechanisms to implement a programming model. For example, one can build a shared memory model that gives the application programmer a single, shared address space.

The implementation of a combining tree¹⁸ illustrates the use of the MDP mechanisms. The combining tree (Figure 3) consists of a number of nodes each containing a value, a count, and a pointer to a parent node.

We initialize the value to zero and the count to the number of inputs expected. To sum the values of a number of parallel processes, each node sends a COMBINE message containing the result of its process to a combining node. When the messages arrive, the processor containing the combining node creates a task to execute the COMBINE routine. The routine adds the message value to the node's value and decrements the count. When the count reaches zero, the node sends a COMBINE message to the node's parent.

Communication. The MDP supports communication using a SEND instruction for message formatting, a fast network for delivery, automatic message buffering, and task creation upon message arrival.

A series of SEND instructions carries a message of arbitrary length to any node in the machine. Upon arrival at the receiving node, a hardware queue buffers the message.

When the message reaches the head of the queue, the node dispatches a task to handle the message. The combining tree example uses a pair of SEND instructions to send the COMBINE message to a node. Upon message arrival, the MDP buffers the message and creates a task to execute the COMBINE routine.

Synchronization. The MDP synchronizes using message dispatch and presence tags on all states. Because each message arrival dispatches a process, messages can signal events on remote nodes. For example, in the combining tree ex-

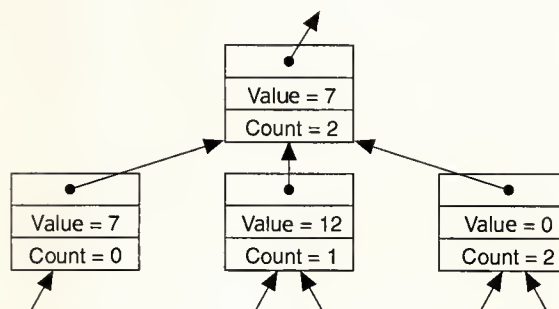


Figure 3. A combining tree sums results produced by a distributed computation. Each node sums the input values as they arrive and then passes a result message to its parent.

ample, each COMBINE message signals its own arrival and initiates the COMBINE routine.

In response to an arriving message, the processor may set presence tags for task synchronization. For example, access to the value produced by the combining tree may be synchronized by initially tagging as empty the location that will hold this value. An attempt to read this location before the combining tree had written it would raise an exception and suspend the reading task until the root of the tree writes the value. Synchronization on data availability in this manner is quite common in many parallel programs.

Naming. The MDP supports naming with segmented memory management and translation instructions. In the combining tree example, the MDP allocates a memory segment to hold the state of each combining node. Using a segment descriptor, it relocates and protects accesses to the node. To make combining nodes relocatable across processing nodes, the MDP translates a node's virtual address to find the processing node where it resides. Upon reaching this node, a second translation locates the segment descriptor for the combining node.

Instruction set architecture

The MDP extends a conventional microprocessor instruction set architecture (ISA) with instructions to support parallel processing. Specifically, the MDP provides efficient hardware mechanisms for communication, synchronization, and naming. Although we describe here the MDP ISA, with particular emphasis on these mechanisms, readers can find more details in Dally et al.¹⁹

Register set. The MDP provides separate register sets to support rapid switching between three execution levels: background, priority 0 (P0), and priority 1 (P1). The MDP executes at the background level when no messages are pending. Each arriving message creates a task and initiates execution at P0 or P1, depending on the message's priority. The MDP executes the highest priority task at any point in time. The arrival of a P1 message while the MDP is executing a P0 task causes the MDP to switch execution levels (and thus register sets). When the P1 task completes, the MDP resumes execution at P0 by switching to the P0 register set that holds the register state of the suspended task.

The register set at each priority level includes

- four general-purpose data registers, R0-R3,
- four address registers, A0-A3,
- four ID registers, ID0-ID3, and
- one instruction pointer, IP.

The background register set does not include ID registers. They only exist at P0 and P1.

Most instructions operate on the general registers R0-R3. Each address register A0-A3 contains a segment descriptor

consisting of a base and a length field. Memory addresses are specified by an offset and an address register. For example, the operands [R0, A1] and [3, A2] specify an indexed access to the segment described by A1 and a displacement of three words into A2's segment.

ID registers usually hold object IDs. The instruction pointer includes process status bits that control virtual addressing, type checking, and fault handling. Placing these bits in the instruction pointer enables control and execution states to change by loading a single register. The relatively small size of each register set facilitates quick task switching within an execution level.

Tags. The MDP uses tags for type checking and synchronization. Every 36-bit word of register and memory state holds a 32-bit value and a 4-bit tag that indicates the type of the value. Tag values are defined for primitive user data types (such as symbol, integer, and Boolean) and for system data types, such as IP, Addr (a segment descriptor), and Msg (a message header). Four tag values are user-definable. If type checking is enabled, the MDP checks operand tags to determine which form of an instruction to execute. It raises an exception if the operands are incompatible with the instruction.

Two tags, Fut and Cfut, support intertask synchronization. A Cfut tag initially marks a location empty. When a task produces the value for the location, it overwrites the Cfut with the final value and tag. Any attempt to read from the location before the value is produced invokes the Cfut fault handler, which typically suspends the reading task until the location is written. Fut is used for global synchronization, and Cfut for local.

Hardware support for tags makes software more efficient and robust. A program can perform an operation without checking whether operands are present or of the correct type. For normal cases in which no fault occurs, execution proceeds faster than if special test and branch instructions were required to check for type and presence. Only exceptional cases incur the overhead of running a fault handler.

Instructions. The MDP executes 17-bit, fixed-format, three-address instructions with the format shown in Figure 4. Each instruction specifies an operation, two general register operands, and a third operand that may be a register, a memory location, or a constant. Two 17-bit instructions fit into each 36-bit word. Any instruction stream word not tagged as an

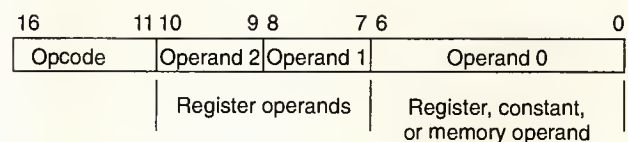


Figure 4. MDP instruction format.

General movement and type instructions				
READ	WRITE	READR	WRITER	RTAG
WTAG	LDIP	LDIPR	CHECK	
Arithmetic and logic instructions				
CARRY	ADD	SUB	MULH	MUL
ASH	LSH	ROT	AND	OR
XOR	FFB	NOT	NEG	LT
LE	GE	GT	EQUAL	NEQUAL
EQ	NEQ			
Network instructions				
SEND	SENDE	SEND2	SEND2E	
Associative lookup table instructions				
XLATE	ENTER	PROBE		
Special instructions				
NOP	INVAL	SUSPEND	CALL	
Branches				
BR	BNIL	BNNIL	BF	BT
BZ	BNZ			

Figure 5. Six categories of MDP instructions.

instruction is loaded as a constant into register R0. This provides a very efficient means to load arbitrary 36-bit constants. Figure 5 summarizes the MDP instruction set by category.

Naming. The MDP supports naming via translation instructions and segmented addressing. Addressing memory through segment descriptors permits arbitrary size objects to be relocated and protected. The ENTER instruction enters an arbitrary translation from a 36-bit key to a 36-bit data value in a set-associative cache (translation table) mapped into the on-chip memory. The XLATE instruction looks up the data value (if any) associated with a key. These instructions can translate an object's name into a physical segment descriptor or a node number to support a global virtual address space.

Communication. The MDP provides hardware support for end-to-end message delivery including formatting, injection, delivery, buffer allocation, buffering, and task scheduling.

An MDP transmits a message using a series of SEND instructions, each of which injects one or two words into the network at either priority 0 or 1. Figure 6 shows a typical

```

SEND    R0,0      ; send net address (priority 0)
SEND2   R1,R2,0   ; header and receiver (priority 0)
SEND2E  R3,[3,A3],0 ; selector and continuation -
                    ; end msg. (priority 0)

```

Figure 6. MDP assembly code to send a four-word message uses three variants of the SEND instruction.

message send. The first SEND instruction reads the absolute address of the destination node in <X,Y,Z> format from R0 and forwards it to the network hardware. The SEND2 instruction reads the first two words of the message out of registers R1 and R2 and enqueues them for transmission. The final instruction enqueues two additional words of data, one from R3, and one from memory. The use of the SEND2E instruction marks the end of the message and causes it to be transmitted into the network. This sequence executes in four clock cycles (250 ns).

The network delivers an injected message to the destination node, as described later. At the destination, a hardware-managed, FIFO queue in the internal RAM of the MDP buffers the message. Separate queues exist for P0 and P1 messages.

Task scheduling. When a message reaches the head of the highest priority nonempty queue, the MDP creates a task to handle it by changing the thread of control and creating a new addressing environment, as shown in Figure 7. Every message header contains a message opcode and the message length. The MDP loads the message opcode into the instruction pointer to start a new thread of control. The length field and the queue head create a message segment descriptor (automatically written to A3) that represents the initial addressing environment for the task. The message handler code may open additional segments by translating object IDs in the message into segment descriptors. Creating a task to handle a message takes three cycles.

The dispatch mechanism directly processes messages requiring low latency (for example, combining and forwarding). Other messages, such as a remote procedure call, specify a handler that locates the required method (using the translation mechanism described earlier) and then transfers control to the method.

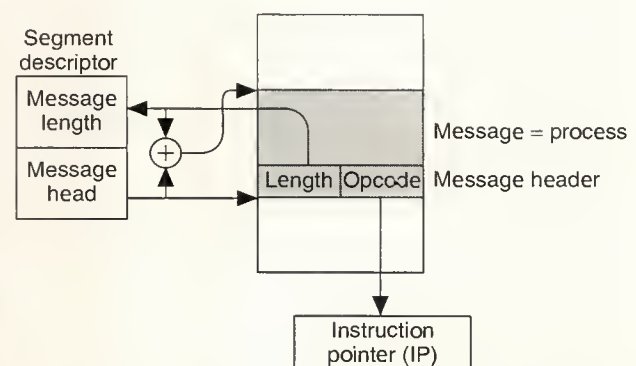


Figure 7. Message dispatch. In three clock cycles, a node creates a new task by setting the instruction pointer to change the thread of control and creating a message segment to provide the initial addressing environment.

```

MOVE  [1,A3],R0    ; get method ID
XLATE  R0,A0        ; translate to segment descriptor
LDIP   INITIAL_IP  ; load instruction pointer to
                    ; transfer control to method

```

Figure 8. MDP assembly code for the CALL message.

For example, Figure 8 shows the CALL handler code handling a remote procedure call. Figure 9 depicts the execution of the handler. The first instruction gets the method ID (offset one word into the message segment referenced by A3). The next instruction translates this method ID into a segment descriptor for the method and places this descriptor in A0. In one of its operating modes, the MDP can use A0 as a pointer to a segment of code and IP as an index into that segment. This allows code to be easily relocated at runtime. The final instruction of the CALL handler transfers control to the method by loading the IP with a short integer offset. Thereafter the MDP will fetch instructions from the called method.

The method code may then read in arguments from the message queue. The XLATE instruction translates argument object identifiers to physical memory base/length pairs. If the method needs space to store local state, it may create a context object. When the method finishes executing, or when it needs to wait for a reply, it executes a SUSPEND instruction, which dequeues its message and passes control to the next message in the queue.

An example of a direct message handler is the COMBINE routine shown in Figure 3. Figure 10 displays the code for this routine. If the node is idle, execution of this routine begins three cycles after message arrival. The routine loads the combining node pointer and value from the message, performs the required add and decrement, and, if Count reaches zero, sends a message to its parent.

This 12-instruction routine executes in 21 cycles. It demonstrates several ways in which the MDP's communication mechanism reduces the overhead of message passing to the point where

it can perform simple operations, such as combining. These ways include the following:

- The MDP hardware dispatches the COMBINE task by setting the instruction pointer to COMBINE and initializing message pointer A3 to allow direct access to message words. This avoids the overhead otherwise associated with control transfer and with setting up an addressing environment.
- The two SEND instructions transmit the four-word mes-

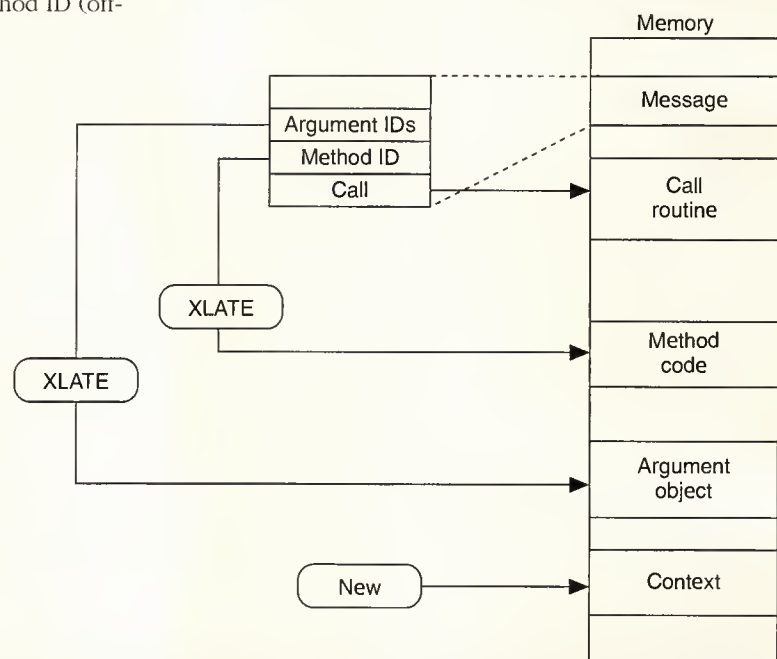


Figure 9. The CALL message invokes a method by translating the method identifier to find the code, creating a context (if necessary) to hold local state, and translating argument identifiers to locate arguments.

```

COMBINE:  MOVE    [1,A3], COMB      ; get node pointer from msg
          MOVE    [2,A3], R1       ; get value from msg
          ADD     R1, COMB.VALUE, R1
          MOVE    R1, COMB.VALUE    ; store result
          MOVE    COMB.COUNT, R2    ; get Count
          ADD     R2, -1, R2
          MOVE    R2, COMB.COUNT    ; store decremented Count
          BNZ     R2, DONE
          MOVE    HEADER,R0         ; get message header
          SEND2   COMB.PARENT_NODE, R0 ; send message to parent
          SEND2E  COMB.PARENT, R1   ; with value
DONE:     SUSPEND

```

Figure 10. MDP assembly code for the combining tree example.

sage to the parent task. The message transmits directly from register and memory variables with no need to first format it in memory.

- The SUSPEND instruction terminates the task and simultaneously dequeues the message. If another message is pending in the queue, the processor dispatches a task to handle it two cycles after the execution of the SUSPEND instruction.

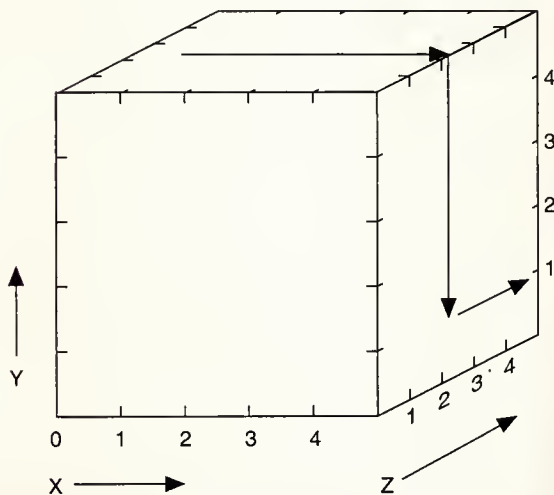


Figure 11. The J-Machine network is a 3D mesh or k -ary 3-cube. The network performs e-cube or destination tag routing. Messages route in each dimension in turn to the proper coordinate in that dimension. In this figure, a message routes from (1,5,2) to (5,1,4), routing first in X, then Y, then Z.

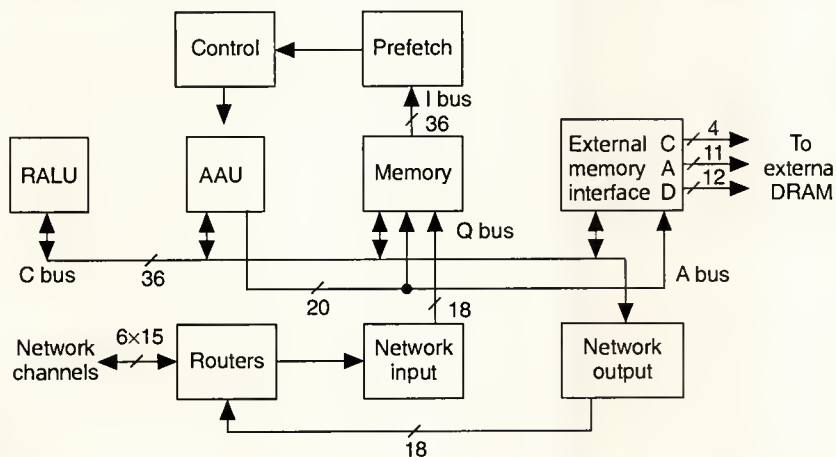


Figure 12. MDP block diagram.

Network architecture

The MDP contains a network interface and a router that support a communication network closely integrated with the processor. In a J-Machine composed of MDPs, the network provides end-to-end message delivery with low latency (less than $2 \mu\text{s}$ in a 4,096-node network) and high bandwidth (288 Mbits per second per channel). Message delivery occurs entirely within the routers of the machine and consumes no processor or memory resources at intermediate nodes.

Structure. The J-Machine network is a 3D grid, with two-way channels, dimension-order routing, and blocking flow control. (See Figure 11.) Addressing limits the size of the network to 65,536 nodes ($32 \times 32 \times 64$). Our initial prototype is a 1,024-node machine ($8 \times 8 \times 16$). The faces of the network cube are open for use as I/O ports to the machine. Each channel can sustain a data rate of 288 Mbits. All three dimensions may operate simultaneously for an aggregate data rate of 864 Mbits per node.

Three modules, shown in Figure 12, compose the network logic. The network output module buffers words and injects them into the network. The three routers, one for each dimension of the network, route messages from node to node. The network input module reassembles messages at their destination and buffers them into a message queue. We describe more details of implementation in the next section.

Engineering. We chose the 3D mesh topology of the J-Machine network as the most efficient arrangement subject to constraints of wiring density and component pinout.⁷ These constraints set the width of the six bidirectional channels per MDP node at 9 data bits plus 6 control bits. We built the J-Machine as a stack of boards with dense board-to-board interconnections to implement the 3D network with short wires.

The MDP breaks with the tradition of asynchronous network routers by implementing a synchronous router.^{16,17} This router operates at twice the rate of the processor, sending a pair of 9-bit *phits* between nodes each 62.5-ns processor cycle (A phit is a physical digit, the width of the physical channel. A pair of phits form a *flit*, or flow-control digit, the granularity of flow control in the network. An 18-bit flit is half an MDP data word.)

Each of the six bidirectional channels can be turned around on alternate cycles with no contention penalty. A novel pad design tolerates clock skew between routers and eliminates the potential for conduction overlap when the channel reverses direction.²⁰ Messages route through the network with a latency of one 62.5-ns processor cycle per hop. Thus, message latency T is given by

$$T = T_c (2L + D),$$

where T_c is the processor cycle time, L is message length in words, and D is the distance (number of nodes) a message must traverse. For example, in a 1,024-node machine, an $L=6$ word message to a random destination traverses an average of $D=10$ nodes for a latency of $T=22$ cycles or 1.4 μ s. The bisection bandwidth (the bandwidth across a plane dividing the machine into two equal halves) of a 1,024-node machine is 18.4 Gbps. The aggregate bandwidth of the network channels is 864 Gbps, and the I/O bandwidth is 184 Gbps.

Routing and flow control. The J-Machine uses deterministic dimension order routing, also called e-cube routing. As shown in Figure 11, all messages route first in the X dimension, then in Y , then in Z . Since messages route in dimension order and messages running in opposite directions along the same dimension do not block, we avoid resource cycles, and leave the network provably deadlock free.²¹

Table 1 lists the format of a message. The first three flits of the message contain the X , Y , and Z addresses. Each node along the path compares the address in the head flit of the message with the node's index in the current dimension. If the two indices match, the node strips the head flit off the message and routes the rest to the next dimension. The MDP's network output node formats the address flits of the message. It also precomputes the direction (positive or negative) the message must travel along each dimension, setting additional bits in the address flits. This reduces the latency and complexity of the router nodes.

The network uses blocking flow control to resolve contention for a physical channel (see Figure 13). When a message arrives at a router path already in use by a message of the same priority, it is blocked. The blocked message compresses

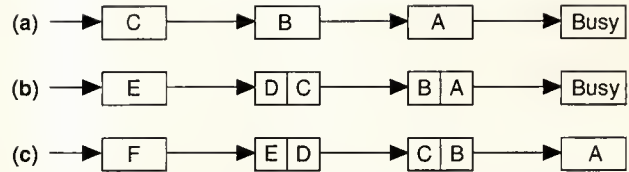


Figure 13. The J-Machine network performs blocking flow control with two stages of queuing per node. Message arrives at busy channel (a). Message becomes compressed by queuing (b). Channel is available; message continues advancing (c).

into routers along its path, occupying one node per word (two flits) of the message. When the blockage clears, the message uncompresses and proceeds to its destination, at a rate of one hop per cycle.

Two priorities of messages share the physical wires, but use completely separate buffers and routing logic. This allows priority 1 messages to proceed through blockages at priority 0. Without this ability, the system could not redistribute data that has caused hot spots in the network.

MDP implementation

Figure 12 shows the major subsystems in the MDP. The chip includes a conventional microprocessor with prefetch, control, register file and ALU (RALU), and memory blocks. The communication system comprises the routers and network input and output interfaces. The address arithmetic unit (AAU) provides addressing functions. The MDP also includes a DRAM interface, control block, and diagnostic interface.

Communication subsystem. The communication subsystem contains the network output, the network input, and the routers. The network output block buffers messages from the registers or memory and injects them into the network. A FIFO buffer matches the speed of message transmission to the network. On each SEND instruction, the MDP transfers one or two words to its FIFO. When the message is complete, or the eight-word buffer is full, the buffer launches the message into the network. In cases where the MDP cannot send message words as fast as the network can transmit them, the FIFO prevents bubbles (absence of words) from entering the network pipeline and degrading performance.

The network input module transfers messages from the network to the MDP's memory. Data from the network arrive in 18-bit flits, which are composed into a four-word queue row buffer. When the QRB fills, it writes its contents to the on-chip memory in one cycle. Writing memory a row (4×36 bits) at a time reduces the number of memory cycles consumed by the network, leaving more memory bandwidth for the CPU.

The routers form the switches in a J-Machine network and

Table 1. A typical message in the J-Machine.

Flit	Contents	Remark
1	5:+	X address
2	1:-	Y address
3	4:+	Z address
4	MSG: 00	Method to call
5	00440	
6	INT: 00	Argument to method
7	0023	
8	INT: 00	Reply address
9	<1:5:2> T	

The first three flits contain the destination address. The final flit in the message is marked as the tail.

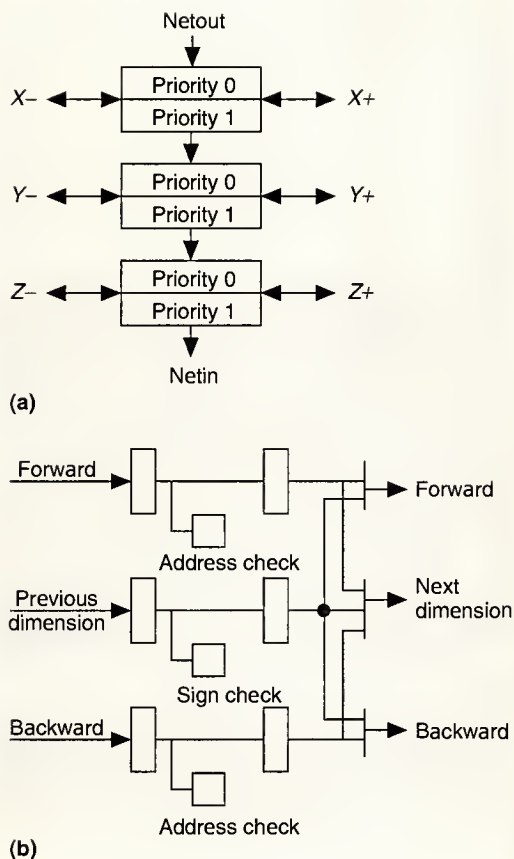


Figure 14. Block diagram of the routers. The two priorities per dimension are completely separate except where they share physical channels (a). Each priority contains forward, reverse, and previous to next dimension datapaths (b).

deliver messages to their destinations. As shown in Figure 14a, the MDP contains three independent routers, one for each bidirectional dimension of the network. Each router contains two separate virtual networks with different priorities that share the same physical channels. The priority 1 network can preempt the wires even if the priority 0 network is congested or jammed.

Each of the 18 router paths contains buffers, comparators, and output arbitration (Figure 14b). On each data path, a comparator compares the lead flit, which contains the destination's address in this dimension, to the node coordinate. If the head flit does not match, the message continues in the current direction. Otherwise the message is routed to the next dimension. Messages entering the dimension compete with messages continuing in the dimension at a two-to-one switch. Once a message is granted this switch, any other

input is locked out for the duration of the message. Once the head flit of the message has set up the route, subsequent flits follow directly behind it.

Address arithmetic unit. The AAU, the largest logic block in the MDP, performs all functions associated with memory addressing. To support naming and relocation, the AAU contains the address and ID registers. It protects memory accesses and implements the translation instructions. Each memory reference is offset by the selected address register's base field and checked against its length field. An attempt to access through an invalid address register (which may occur when an object relocates) or to access beyond the end of an object raises an exception. A translation base/mask register defines an area of memory to be a two-way, set-associative translation buffer used by the XLATE, PROBE, and ENTER instructions. The AAU hashes the keys used to access this table using an exclusive-Or network to improve hit rate in the translation buffer.

The AAU maintains two queues to buffer incoming messages and schedule the associated tasks. Associated with each queue are a queue base/mask (QBM) and a queue head/length (QHL) register. (See Figure 15.) The QBM registers define the position and length in main memory of the message queues. Queues are circular, so messages at the end of the queue wrap around to the beginning. The QHL registers point to the beginning of the first message in the queue and its length field encompasses exactly all of the messages currently in the queue. When the MDP dispatches a task to handle a message, it loads the A3 register with a segment descriptor for the message. The processor dispatches a task as soon as the first four words of a message are written. If the task attempts to read a word of the message which has not yet arrived, a special Early fault occurs.

Layout. Figure 16 shows a floor plan of the chip with a die photograph for comparison. Table 2 breaks down the area usage.

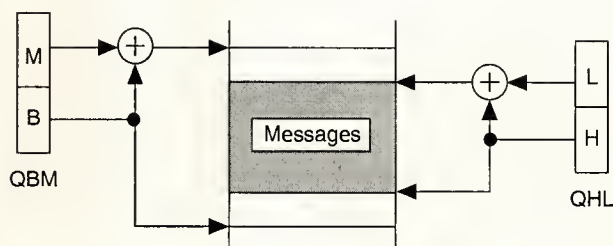


Figure 15. The AAU maintains the queue base/mask (QBM) registers, which specify the location of the message queues in main memory, and the queue head/length (QHL) registers, which specify the beginning and end of the messages received in each queue. Figure shows only one queue.

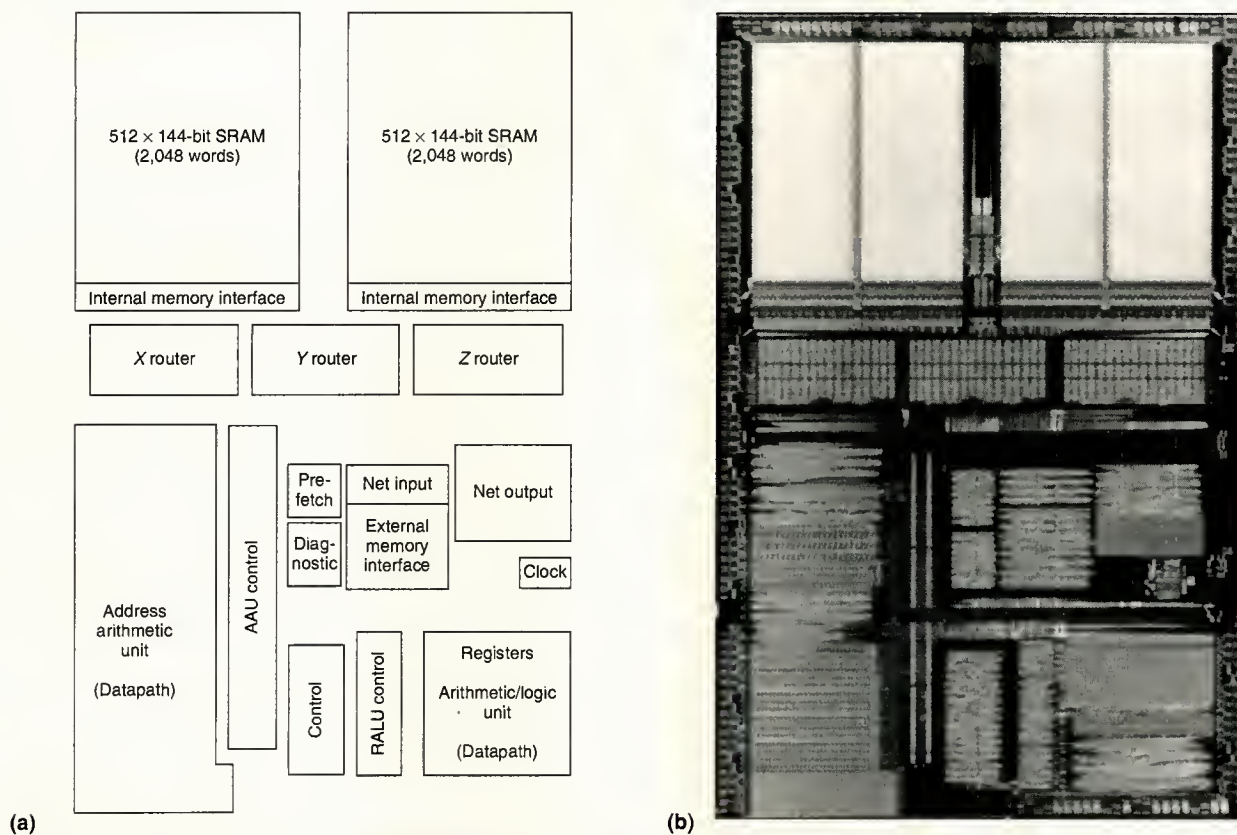


Figure 16. MDP chip floor plan (a) and die photograph (b).

Methodology. We implemented the MDP using Intel standard cells except for the on-chip RAM, clock generator, and pads. Using standard cells sacrificed a factor of three to four in area and two to three in performance over what would be possible with full-custom design. The advantage was a significant increase in productivity which was essential to completing the chip successfully with our small design team.

The 700 or so sheets of schematics drafted at MIT used 35,000 standard cells containing 210,000 transistors. (The remaining 890,000 devices are contained in the full custom portions of the chip, mostly in the RAM.) We sent these schematics to Intel for layout. Designers laid out many of the data paths by hand to exploit the regularity of the design. Automatic place and route CAD tools laid out the less regular collections of logic.

We began architecture studies leading to the MDP in October 1986. Work on the RTL model of the microarchitecture began in June 1988, and schematic entry at MIT started that November. The task of translating schematics into layout commenced in June 1989, and we finished the layout in December 1990. We received first silicon in June 1991 and were running programs on it within a few hours.

Table 2. Chip area breakdown.

Module	Dimensions (mm)	Area (mm ²)	Transistors (×10 ³)
AAU	3.7 × 7.0	25.9	75.0
RALU	3.7 × 2.9	10.7	39.0
Diagnostic	0.9 × 1.1	1.0	3.7
Prefetch	0.9 × 1.1	1.0	3.2
Control	1.1 × 2.6	2.9	8.7
Internal memory interface	7.8 × 0.5	3.9	13.0
External memory interface	1.6 × 1.8	2.9	9.0
Net input	1.8 × 0.7	1.3	4.4
Net output	2.1 × 1.8	3.8	18.0
Routers	8.4 × 1.3	10.9	29.0
RAM	8.8 × 4.9	43.1	880.0
Clock	0.7 × 0.8	0.6	0.1
Pads	50.5 × 0.2	8.4	2.6
Full chip	10.2 × 15.0	153.0*	1,087.0

* Includes wiring between modules.

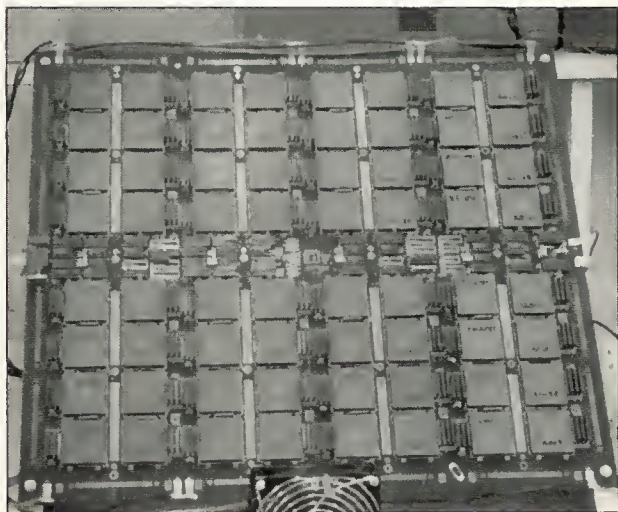


Figure 17. Photograph of 64-node J-Machine system.

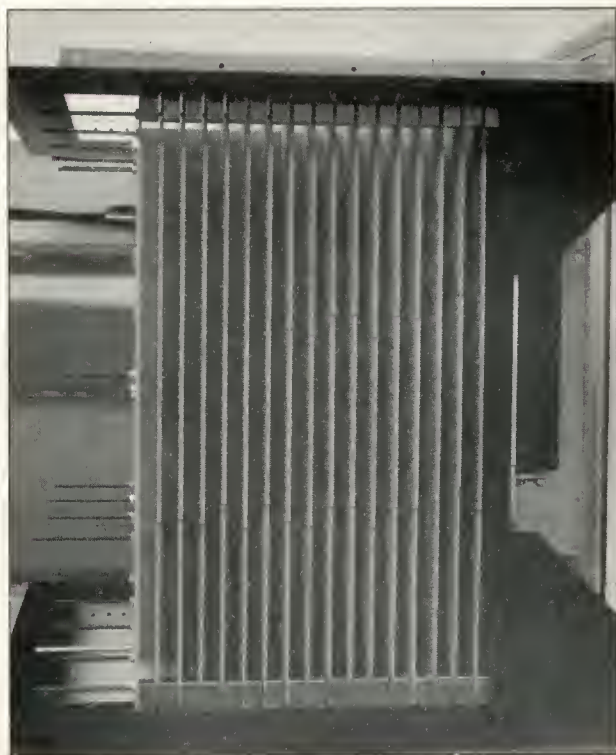


Figure 18. A 1,024-node J-Machine chassis.

Although we thoroughly simulated the logic design, we have uncovered 12 bugs while running our validation tests and applications on the hardware. Some of these bugs have simple software work-arounds, but for performance reasons we sent a second revision of the layout with modified control logic and some metal fixes for fabrication in January of this year. We plan to use several thousand of these chips to build research multicomputers at MIT.

System design. Figure 17 shows a photograph of a 64-node J-Machine processor board measuring 20.5 in. \times 24 in. Each node consists of an MDP chip (in a 168-pin grid array package) and three 4-Mbit DRAMs. Each pair of nodes shares a set of elastomeric connectors to communicate with the corresponding nodes on the boards above or below the board in a stack. A total of 32 elastomeric connectors held in four connector holders provide 2,240 electrical connections between adjacent boards. Of these connections, 960 are used for signalling and the remaining are ground returns. No power is supplied through the elastomers. Bus bars supply power and ground directly to each board. The center area of the board contains the final stage of the clock distribution network, along with diagnostic fan-out, multiplexing logic, and temperature and airflow monitors.

Figure 18 shows a photograph of our chassis for a 1,024-node system. The chassis contain a stack of 16 processor boards, power supplies, and distribution bus bars. Twenty tie rods bind the boards and compress the elastomer connectors. A 4,096-node system can be built by combining four chassis. Each stack connects to its neighboring stacks by 128 (16×8) short, 60-pin, ribbon cables—one for each pair of nodes on the periphery. Each vertical pair of stacks shares a 3,000 cu ft/min. blower for cooling.

In addition to the processor board and chassis, we have also designed a diagnostic interface board and are designing a SCSI disk interface, a distributed graphics frame buffer, and an S-bus interface. Noakes and Dally²² offer more details of the J-Machine system design.

Software

We intended the J-Machine as a platform for software experiments in fine-grain, parallel programming. To this end, we have implemented and are studying software systems for different fine-grain programming models. Fine-grain programs typically execute from 10 to 100 instructions between communication and synchronization actions. Reducing the grain size of a program increases both the potential speedup due to parallel execution and the potential overhead associated with parallelism. Special hardware mechanisms to reduce the overhead due to communication, process switching, synchronization, and multithreading are therefore central to the design of the MDP. Software issues such as load balancing, scheduling, and locality remain open questions and are the focus of current research efforts.

```

(defmethod Size-Of-Tree Pair ()
  (+ (Size-Of-Tree Left)
     (Size-Of-Tree Right)))
(defmethod Size-Of-Tree Object ()
  1)
(defmethod Size-Of-Tree Null ()
  0)

```

Figure 19. Concurrent Smalltalk source to compute Size-Of-Tree. Method definitions specify the class to which they apply. The class Pair contains two elements, Left and Right, each of which may hold an Object or another Pair.

A parallel processor creates programming challenges. It is difficult to extract the fine-grain parallelism needed from stock programs written in C or Fortran. Instead of concentrating on extracting parallelism from existing programs (an active and interesting area for many parallel programming researchers) or on adapting sequential languages for the parallel domain, we focus on languages where the expression of fine-grain parallelism is much cleaner. To date, we have implemented two languages on the J-Machine: the actor language Concurrent Smalltalk and the dataflow language Id.

Concurrent Smalltalk. CST²³ is a parallel, object-oriented, programming language (based on the Actor model⁵) with asynchronous message send and distributed objects. Its syntax is similar to that of Lisp or Scheme. It performs method or function invocation by sending a message to the first argument of the method. The message contains the method selector and the rest of the arguments.

Functions and methods in the language are compiled into MDP assembly code by an optimizing compiler, called Optimist, and assisted at runtime by a small kernel called Cosmos.

MODULE	OBJ:Selector.Size_Of_Tree	
DC	Copyable class_Selector	; Identify properties of
		; Size_Of_Tree selector
DC	OBJ:Selector.Size_Of_Tree	; Store own ID inside selector
DC	3	; Number of functions
DC	CLASS:Object	; Class identifier for Object
DC	{function.Size_Of_Tree}	; Function for class Object
DC	CLASS:Null	; Class identifier for Null
DC	{function.Size_Of_Tree_1}	; Function for class Null
DC	CLASS:Pair	; Class identifier for Pair
DC	{function.Size_Of_Tree_2}	; Function for class Pair

Figure 20. Selector object generated by the example program.

Cosmos provides a global virtual name space, object-based memory management, support for distributed objects, and low-overhead context switching. Its memory management system provides fast, transparent access to storage distributed across the machine. Cosmos efficiently supports fine-grain concurrent computation in which tasks are very short (40 user instructions) and data objects are very small (eight words). The CST compiler and the Cosmos runtime system also provide floating-point arithmetic, simple arrays, and garbage collection for CST programs. Cosmos manages contexts, futures, and objects, and therefore plays an important role in providing services that exploit the communication, synchronization, and naming mechanisms of the J-Machine.

Figure 19 shows a small sample program defining the Size-Of-Tree method for three object types: a Pair, the Null object, and a generic object. When called on a Lisp-style tree, these methods return the number of generic objects stored in the tree. For example, when called on the tree '(1 2 3)(4 5 6)(7 8 9)), Size-Of-Tree returns the value 9. Note that since Pair and Null are subclasses of Object, their more specific methods are selected when Size-Of-Tree is invoked on their types.

When Optimist, the CST compiler, compiles this example program, it defines a selector object and three function objects. The selector object (shown in Figure 20) lists the type and function correspondence. When a method applies to an object, Cosmos examines the object type and locates the appropriate function in the selector object. The MDP then invokes this function on the object. (In cases where the compiler can infer the type of the object or when the type of objects is explicitly declared, the compiler optimizes a method invocation directly to the correct function invocation.) The compiler marks the selector object as copyable, and Cosmos maintains it like any other object.

Figure 21 shows the compiled code for the function for the class Pair. When a method applies to a particular object, Cosmos examines the object class and the selector object, and chooses the correct function to invoke.

The function first does an XLATE operation to get the address of the Pair and uses that address to get the object ID for Left. It then calls Cosmos to find the node where Left exists. The function sends a message to Left that recursively applies the Size-Of-Tree method. It marks the slot that will hold the return value with a Cfut tag. Next, it applies Size-Of-Tree to Right without waiting for the result of the first remote procedure to return. However, when the function attempts to add the two return values, the results will probably not have returned yet. In this case, the ADD instruction will fault trying to add Cfuts, and the MDP will suspend the process, saving its registers into the context.


```

MODULE      OBJ:function.Size_Of_Tree_2

DC          Copyable | class_Function      ; Identify properties of Size_Of_Tree function
DC          {OBJ}:function.Size_Of_Tree_2} ; Store own ID inside function

;; Incoming: A1 points to the context
;;           A3 points to the message

START:
MOVE        [2,A3],R3                      ; Get the Pair's object ID
XLATE       R3,A2                          ; Find the Pair's local address
MOVE        [2,A2],R0                      ; Get the object ID of Left
CALL        objectNode,R1                  ; Find Left's node -> R1
DC          MSG:Apply_Selector              ; Send a message to Left to apply
SEND2       R1,R0                          ; the method specified by the
DC          {OBJ}:Selector.Size_Of_Tree}   ; selector for Size_Of_Tree.
SEND        R0                             ; Includes our context ID
SEND        [2,A2]                         ; and a continuation.
MOVE        5,R0
SEND2E      [1,A1],R0
WTAG        R0,CFUT,R0                    ; Make a future for Left's
MOVE        R0,[5,A1]                     ; result.
MOVE        [3,A2],R0                     ; Do the same for Right as for
CALL        objectNode,R1                 ; Left.
DC          MSG:Apply_Selector
SEND2       R1,R0
DC          {OBJ}:Selector.Size_Of_Tree}
SEND        R0
SEND        [3,A2]
MOVE        6,R0
SEND2E      [1,A1],R0
WTAG        R0,CFUT,R0
MOVE        R0,[6,A1]
MOVE        [6,A1],R2                     ; Do the sum.
ADD         R2,[5,A1],R1
MOVE        [3,A3],R3                     ; Get the continuation for
BNIL        R3,^L001                      ; this context. Reply if
DC          MSG:Reply                       ; non-nil.
SEND2       R3,R0
SEND        R3
SEND2E      [4,A3],R1
L001:
SUSPEND
END

```

Figure 21. Compiled code for the Size-Of-Tree function for objects of class Pair.

Assuming this happens, when the MDP receives replies from the methods after writing the value into the future slot, Cosmos checks to see if the process was waiting for that particular future. If so, it reactivates the context. The reactivated function would then sum the two results and forward them to the con-

tinuation specified in the original method invocation.

Let us consider some interesting points:

- If the object of the function is not present or if the translation cache does not have an entry for the object, the

XLATE instruction will fault. Cosmos will find the object and move or copy it to the local node.

- Cosmos maintains functions and selectors like any other immutable object. If they are not present, Cosmos will copy them to the node, a process analogous to a distributed instruction cache.
- If the function were preempted and the object moved or migrated away, Cosmos would invalidate the address registers. Accesses to the object would cause a fault that would attempt to retranslate or reobtain the object.
- The A1 register points to the current context. The context contains storage to hold working variables or, if the context faults, to hold spilled register values. In the example, the futures are constructed in the context, and thus are named *context-future* (Cfut).

This example illustrates some important research questions related to the efficiency of this model of computation.

- When is it better to spawn processes nonlocally rather than locally? This is probably a strong function of the amount of associated overhead. The MDP architecture attempts to reduce this overhead, but algorithms for making this trade-off at compile and runtime still need to be developed and evaluated.
- How should we place objects in the machine, and how should they migrate in order to reduce the overhead of communication?
- In some cases, the amount of parallelism grows much larger than the machine can handle. We need to study how we can effectively and automatically throttle the parallelism created by the machine when it becomes saturated.

Horwat discusses these issues, and others related to the efficiency of programming fine-grain, parallel processors in more detail.²³

Dataflow implementation. Id is a functional programming language originally designed for dataflow architectures.²⁴ The Id compiler converts an Id program into a dataflow graph, in which nodes represent operators and arcs represent dependencies. Originally, researchers executed these dataflow graphs directly on specialized dataflow machines. More recently, they have begun compiling dataflow graphs to run on general-purpose parallel machines.²⁵ Dataflow programs suit large parallel computers, because the abundance of fine-grain tasks—each of which can be as small as a single dataflow operator—makes it easy to mask communication latency with task switches. Conversely, the J-Machine's fine-grain mechanisms make it an excellent target for dataflow programs.

We experimented with several methods of executing dataflow programs on the J-Machine.²⁶ The simplest of the systems translates each node of the dataflow graph into a

sequence of MDP instructions. A dataflow node with two inputs takes 20 MDP instructions to simulate. To do so, it stores the first data value, matches it with the second value when it arrives, performs the dataflow operation, and sends the resulting value to two destinations. This process uses the Cfut tag and fault handler.

A more efficient approach increases the granularity of each task to reduce scheduling overhead. We are building a system on top of the Berkeley TAM project²⁵ that addresses the inefficiencies of our earlier systems.

WE BUILT THE MDP TO DEMONSTRATE THE UTILITY of general-purpose communication, synchronization, and naming mechanisms in a multicomputer building block. Its mechanisms efficiently support dataflow²⁶ and object-oriented programming²³ models using a global name space. The use of a few simple mechanisms provides orders of magnitude lower communication and synchronization overhead than is possible with multicomputers built from off-the-shelf microprocessors. Its communication and synchronization performance competes with processing nodes specialized to a single model of computation, such as iWarp¹⁵ (systolic) or the transputer¹³ (communicating sequential processes).

Computers built from fine-grain processing nodes, such as the MDP, consisting of a small but powerful processor and a small memory, are more cost-effective than those built from fewer coarse-grain nodes. Fine-grain nodes devote a larger fraction of their silicon area to processing and have higher arithmetic, memory, and communication bandwidth per unit cost. Large-scale parallel machines built from fine-grain processors have a larger total amount of memory within a given latency of a processor. An efficient network design provides global memory latency and bandwidth competitive with coarse-grain machines.

The MDP is a component for building scalable computer systems. It is useful in configurations ranging from one node to 65,536 nodes. A 128-node Jellybean Machine is currently operational and resources are in place to build several more machines, including a 1,024-node system at MIT and machines at a number of other research institutions.

The MDP project demonstrated the feasibility of building experimental computer systems with limited resources. By concentrating on the novel mechanisms of the MDP and keeping the design simple and modest in other respects, we completed the design of the chip, its system-level hardware, and several programming systems with a handful (less than eight)

of graduate students and engineers in two and a half years.

With the MDP we have begun exploring mechanisms for parallel computers. Much work remains to be done to tune the MDP's mechanisms and compare them to alternatives. The demands of parallel software that drive these mechanisms are very different from the demands placed on sequential computers. We find the design of mechanisms for parallel computers particularly challenging because no well-established parallel benchmarks exist. Additionally, most parallel programs are very biased by the mechanisms (or lack thereof) of the machines for which they were initially written.

Our software studies have suggested improvements that could be made to the MDP. More registers and better mapping mechanisms would be useful. MDP's conservative implementation leaves opportunities for streamlining, by decreasing the cycle time and number of clocks per instruction. A commercial, custom VLSI product based on the architectural mechanisms in the MDP is very plausible.

As technology scales, we can put many powerful processing units on one chip. An interesting direction for further research is the extension of the MDP mechanisms to control intranode as well as internode concurrency. The MIT M-Machine project, now in its early phase, takes this approach. It employs a processor-coupling mechanism to allow local processors to interact with single-cycle latency. ■

Acknowledgments

The MDP was built as part of a collaborative project between MIT and Intel Corporation. The Defense Advanced Research Projects Agency partially funded the project, under contracts N00014-88K-0738 and N00014-91-J-1698. Funding also came from a National Science Foundation Presidential Young Investigator Award, grant MIP-8657531, with matching funds from General Electric Corporation, IBM Corporation, and AT&T Corporation. Intel Corporation provided technical support and personnel.

Andrew Chien, Waldemar Horwat, and D. Scott Wills helped develop the MDP architecture. Scott Fuman, Shaun Kaneshiro, Ellen Spertus, Brian Totty, and Deborah Wallach were instrumental in developing the J-Machine system software. Salim Ahmed, Paul Carrick, Steve Lear, Ted Nguyen, and Mark Vestrich of Intel Corporation provided engineering and management support. Discussions with Chuck Seitz, Steve Ward, Anant Agarwal, and Tom Knight were helpful during the early stages of the J-Machine project. We thank Lisa Sardegna for her help preparing this manuscript.

References

1. W.J. Dally, D.S. Wills, and R. Lethin, "Mechanisms for Parallel Computing," *Proc. NATO Advanced Study Institute on Parallel Computing on Distributed Memory Multiprocessors*, Springer-Verlag, New York, 1991.
2. A. Gottlieb et al., "The NYU Ultracomputer - Designing a MIMD Shared Memory Parallel Computer," *IEEE Trans. Computers*, Vol. C-32, No. 2, Feb. 1983, pp. 175-189.
3. W.D. Hillis and G.L. Steele, "Data Parallel Algorithms," *Comm. of the ACM*, Vol. 29, No. 12, 1986, pp. 1170-1183.
4. A.H. Veen, "Dataflow Machine Architecture," *ACM Computing Surveys*, Vol. 18, No. 4, Dec. 1986, pp. 365-396.
5. G. Agha, "Actors: A Model of Concurrent Computation in Distributed Systems," Tech. Report 844, Artificial Intelligence Laboratory, Massachusetts Inst. of Technology, Cambridge, Mass., 1985.
6. W.C. Athas and C.L. Seitz, "Multicomputers: Message-Passing Concurrent Computers," *Computer*, Vol. 21, No. 8, Aug. 1988, pp. 9-24.
7. W.J. Dally, "Performance Analysis of k -ary n -cube Interconnection Networks," *IEEE Trans. Computers*, Vol. 39, No. 6, June 1990, pp. 775-785.
8. R. Arlauskas, "iPSC/2 System: A Second-Generation Hypercube," *Proc. Third Conf. Hypercube Concurrent Computers and Applications*, Assn. for Computing Machinery, New York, 1988, pp. 33-36.
9. J.F. Palmer, "The Ncube Family of Parallel Supercomputers," *Proc. IEEE Int'l Conf. Computer Design*, IEEE CS Press, Los Alamitos, Calif., 1986, p. 107.
10. *Series 2010 Product Description*, Ametek Computer Research Division, Monrovia, Calif., 1987.
11. "Butterfly Parallel Processor Overview," BBN Report 6148, Bolt, Beranek and Newman Advanced Computers, Inc., Cambridge, Mass., Mar. 1986.
12. W.C. Brantley, K.P. McAuliffe, and J. Weiss, "RP3 Processor-Memory Element," *IEEE Trans. Computers*, Vol. C-34, No. 10, Sept. 1985, pp. 782-789.
13. I. Barron et al., "Transputer Does Five or More MIPS Even When Not Used in Parallel," *Electronics*, Nov. 1983, pp. 109-115.
14. C. Lutz et al., "Design of the Mosaic Element," *Proc. MIT Conf. Advanced Research in VLSI*, Artech Books, Dedham, Mass., 1984, pp. 1-10.
15. S. Borkar et al., "iWarp: An Integrated Solution to High-Speed Parallel Computing," *Proc. Supercomputing Conf.*, IEEE CS Press, Nov. 1988, pp. 330-338.
16. W.J. Dally and C.L. Seitz, "The Torus Routing Chip," *Distributed Computing*, Vol. 1, 1986, pp. 187-196.
17. W.J. Dally and P. Song, "Design of a Self-Timed VLSI Multicomputer Communication Controller," *Proc. Int'l Conf. Computer Design*, pp. 230-234. IEEE CS Press, Oct. 1987.
18. P.C. Yew, N.F. Tzeng, and D.H. Lawrie, "Distributing Hot-Spot Addressing in Large-Scale Multiprocessors," *IEEE Trans.*

- Computers*, Vol. C-36, No. 4, Apr. 1987, pp. 388-395.
19. W.J. Dally et al., "Architecture of a Message-Driven Processor," *Proc. 14th Int'l Symp. Computer Architecture*, IEEE CS Press, June 1987, pp. 189-205.
 20. P.R. Nuth, "Router Protocol," internal memo 23, MIT Concurrent VLSI Architecture Group, 1990.
 21. W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers*, Vol. C-36, No. 5, May 1987, pp. 547-553.
 22. M. Noakes and W.J. Dally, "System Design of the J-Machine," *Proc. Sixth MIT Conf. Advanced Research in VLSI*, W.J. Dally, ed., MIT Press, Cambridge, Mass, 1990, pp. 179-194.
 23. W. Horwat, "Concurrent Smalltalk on the Message-Driven Processor," master's thesis, MIT, May 1989.
 24. R.S. Nikhil, *ID Version 88.1 Reference Manual*, Tech. Report 284, Computation Structure Group, MIT, 1988.
 25. D.E. Culler et al., "Fine-Grain Parallelism with Minimal Hardware Support: A Compiler-Controlled Threaded Abstract Machine," *Proc. Fourth Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, 1991, pp. 164-175.
 26. E. Spertus and W.J. Dally, "Experiments with Dataflow on a General-Purpose Parallel Computer," *Proc. Int'l Conf. Parallel Processing*, Aug. 1991, pp. II-231-II-235.



William J. Dally is an associate professor of computer science at the Massachusetts Institute of Technology, where he directs the group building the J-Machine. His research interests include concurrent computing, computer architecture, computer-aided design, and VLSI design.

Dally received a BS degree from Virginia Polytechnic Institute and an MS from Stanford University, both in electrical engineering. He earned his PhD in computer science from the California Institute of Technology.



J.A. Stuart Fiske is a PhD student at MIT. His main research interests include computer architecture, parallel processing, and VLSI design. He received the BE degree in electrical engineering from McGill University, and the MS in electrical engineering and computer science from MIT. Fiske

is a member of the IEEE.



John S. Keen is a PhD student at the MIT AI Lab. He has worked at IBM Canada, Bell Canada, and Intel. His primary research interests include computer architecture, parallel processing, CAD tools, and concurrent database systems.

Keen graduated from the University of Waterloo with a BAsC degree and an MASc degree, both in electrical engineering. He received a National Science and Engineering Research Council of Canada 1967 Scholarship and is a member of the IEEE and Sigma Xi.



Richard A. Lethin is also pursuing his PhD at MIT's AI Lab. Previously, he designed floating-point processor boards for the Trace-200 and Trace-300 VLIW (very long instruction word) mini-supercomputers at Multiflow Computer. His research interests include computer architecture and

artificial intelligence.

Lethin received a BS degree in electrical engineering from Yale College and an MS in electrical engineering and computer science from MIT. He is a Hertz Foundation fellow.



Michael D. Noakes is a member of the research staff of MIT's AI Lab, where he researches parallel computer architecture. Previously he helped develop the Concert multiprocessor at Harris Corporation. His research interests include computer architecture and parallel processing.

Noakes earned a BS degree in electrical engineering from MIT.



Peter R. Nuth is a PhD student in the MIT AI Lab. He helped design MIT's Concert and J-Machine parallel computers. His primary research interests include computer architecture, communication, and parallel processing.

Nuth earned BS, MS, and Engineer's degrees in electrical engineering and computer science from MIT. He belongs to the IEEE, Sigma Xi, and Eta Kappa Nu.



Roy E. Davison, a 25-year veteran of the IC industry, heads Davison Design and Development in Ben Lomond, California. He has designed chips for several manufacturers, including Texas Instruments, Advanced Micro Devices, National Semiconductor, Intel, and Mas Par. His primary research interest is the design of microprocessors and peripherals. He attended Sam Houston State University.



Gregory A. Fyler is a project manager in Intel's Architecture Development Lab in Santa Clara, California. He has managed chip design for several microcontroller and microcomputer products, most recently the MDP project. Prior to joining Intel 15 years ago, he developed calculator chips for Fairchild Camera and Instrument.

Fyler earned a BS degree in electrical engineering and computer science from the University of California, Berkeley. He is a member of the IEEE.

Direct questions regarding this article to William J. Dally, MIT Artificial Intelligence Lab, 545 Technology Square, NE43-620, Cambridge, MA 02139; or via e-mail at billd@ai.mit.edu

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 156

Medium 157

High 158

IEEE COMPUTER SOCIETY PRESS

Now Available !

1992 Spring Supplement PUBLICATIONS CATALOG

CONTAINS:

*A New Selection of
Computer Science Books,
30 New Proceedings,
3 Special Videotapes,
&
Our Full Line of Technical Books
on Software, Computer Graphics,
Networks, Design and Test,
and More !*

Call 1-800-CS-BOOKS
for your copy today

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

IEEE COMPUTER SOCIETY PRESS

ARCHITECTURAL ALTERNATIVES FOR EXPLOITING PARALLELISM

edited by David J. Lilja

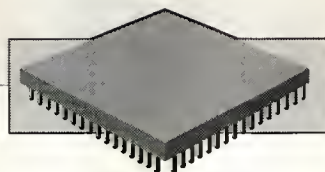
This tutorial surveys the fine-grain parallel architectures that exploit the parallelism available at the instruction set level, the coarse grain parallel architecture that exploit the parallelism available at the loop and subroutine levels, and the single instruction, multiple-data (SIMD) massively parallel architectures that exploit parallelism across large data structures.

It includes over 37 articles that discuss the potential of parallel processing for reducing the execution time of a single program, available parallelism in application programs, pipelined processors, multiple-instruction issue architectures, decoupled access/execute architectures, dataflow processors, shared memory multiprocessors, distributed memory multicomputers, reconfigurable and massively parallel architectures, and comparing parallelism extraction techniques.

464 PAGES, MARCH 1992, ISBN 0-8186-2642-9.
CATALOG # 2642 — \$80.00 MEMBERS \$50.00



1-800-CS-BOOKS
FAX (714) 821-4010
in California (714) 821-8380



Organization of the Motorola 88110 Superscalar RISC Microprocessor

Motorola's second-generation RISC microprocessor employs advanced techniques for exploiting instruction-level parallelism, including superscalar instruction issue, out-of-order instruction completion, speculative execution, dynamic instruction rescheduling, and two parallel, high-bandwidth, on-chip caches. Designed to serve as the central processor in low-cost personal computers and workstations, the 88110 supports demanding graphics and digital signal processing applications.

Keith Diefendorff

Michael Allen

Motorola

Motorola designers conceived of the 88000 RISC (reduced instruction-set computer) architecture to simplify construction of microprocessors capable of exploiting high degrees of instruction-level parallelism without sacrificing clock speed. The designers held architectural complexity to a minimum to eliminate pipeline bottlenecks and remove limitations on concurrent instruction execution.

The 88100/200 is the first implementation of the 88000 architecture. It is a three-chip set, requiring one CPU (88100) chip and two (or more) cache (88200) chips. The CPU's simple scalar design uses multiple concurrent execution units with out-of-order instruction completion to approach a throughput of one instruction per clock cycle.

The second-generation, single-chip 88110 RISC microprocessor employs superscalar instruction issue and out-of-order instruction execution techniques to achieve a throughput greater than one instruction per clock cycle.

Overview

In designing the 88110, we aimed at a general-purpose microprocessor, suitable primarily for use as the central processor in low-cost personal com-

puters and workstation systems. Thus, our design objective was good performance at a given cost, rather than ultimate performance at any cost. We recognized that the personal computer environment is moving toward highly interactive software, user-oriented interfaces, voice and image processing, and advanced graphics and video, all of which would place extremely high demands on integer, floating-point, and graphics processing capabilities. At the same time, we realized the 88110 would have to meet these performance demands while operating with the inexpensive DRAM systems typically found in low-cost personal computers.

To achieve the performance goals set for the 88110, we needed to obtain more parallelism than was achieved in earlier microprocessors. To this end, we decided to use a superscalar microarchitecture to exploit additional instruction-level parallelism. Superscalar machines are distinguished by their ability to dispatch multiple instructions each clock cycle from a conventional linear instruction stream. This approach has shown good speedup on general-purpose applications and was a good match to available CMOS technology. (We believe Agerwala and Cocke coined the superscalar term.¹)

We selected the superscalar approach over

other fine-grain parallelism approaches, such as the vector machine and the VLIW (very long instruction word) approach, because it appeared to be more effective for our intended application. With a limited transistor budget, spending transistors on vector hardware would have meant sacrificing scalar performance and would have yielded a machine that suffered from the Amdahl's Law phenomenon² on general-purpose applications.

The VLIW approach³ would have introduced severe software compatibility restrictions, by exposing hardware parallelism to the object code program and thereby limiting future implementation flexibility. Also, the VLIW speedup, while substantial on code with abundant parallelism (such as scientific applications), is less significant on general-purpose applications. This limited speedup is due in part to the code expansion: The inefficient use of opcode bits to control unused execution units and the aggressive loop unrolling required to schedule the available execution unit parallelism effectively.⁴

The superscalar approach appeared to be a better match to CMOS technology than a superpipelined approach. Superscalar designs rely primarily on spatial parallelism—multiple operations running concurrently on separate hardware—achieved by duplicating hardware resources such as execution units and register file ports. Superpipelined designs, on the other hand, emphasize temporal parallelism—overlapping multiple operations on a common piece of hardware—achieved through more deeply pipelined execution units with faster clock cycles. As a result, superscalar machines generally require more transistors, whereas superpipelined designs require faster transistors and more careful circuit design to minimize the effects of clock skew. Some literature indicates that superscalar and superpipelined machines of the same degree would perform roughly the same.⁵ We felt that CMOS technology generally favors replicating circuitry over increasing clock cycle rates, since CMOS circuit density historically has increased at a much faster rate than circuit speed.

The 88110 microarchitecture, illustrated in Figure 1, employs a symmetrical superscalar instruction dispatch unit, which dispatches two instructions each clock cycle into an array of 10 concurrent execution units. The design implements fully

interlocked pipelines and a precise exception model, but it allows out-of-order instruction completion, some out-of-order instruction issue, and branch prediction with speculative execution past branches.

We optimized each execution unit for low latency: The branch unit uses a branch target instruction cache to reduce branch latency. The integer and graphics units are one-cycle units; the floating-point adder and multiplier are three-cycle, fully pipelined, IEEE extended-precision units. The load/store unit provides fast access to the cache on a hit. But it is also highly buffered to increase tolerance to long memory latency on a miss and to allow dynamic reordering of loads and stores for runtime overlapping of tight loops.

The on-chip caches are organized in a Harvard arrangement, giving the processor simultaneous access to instructions and data. Each 8-Kbyte, two-way, set-associative cache provides 64 bits each clock cycle to its respective unit. The write-back data cache is nonblocking for some types of accesses, and it follows a four-state MESI (modified, exclusive, shared, invalid) protocol⁶ for coherence with other caches in multiprocessor systems. It also supports selective cache bypass and software prefetching from user mode. Two independent, 40-entry, fully associative address translation look-aside buffers (TLBs) support a demand-paged virtual

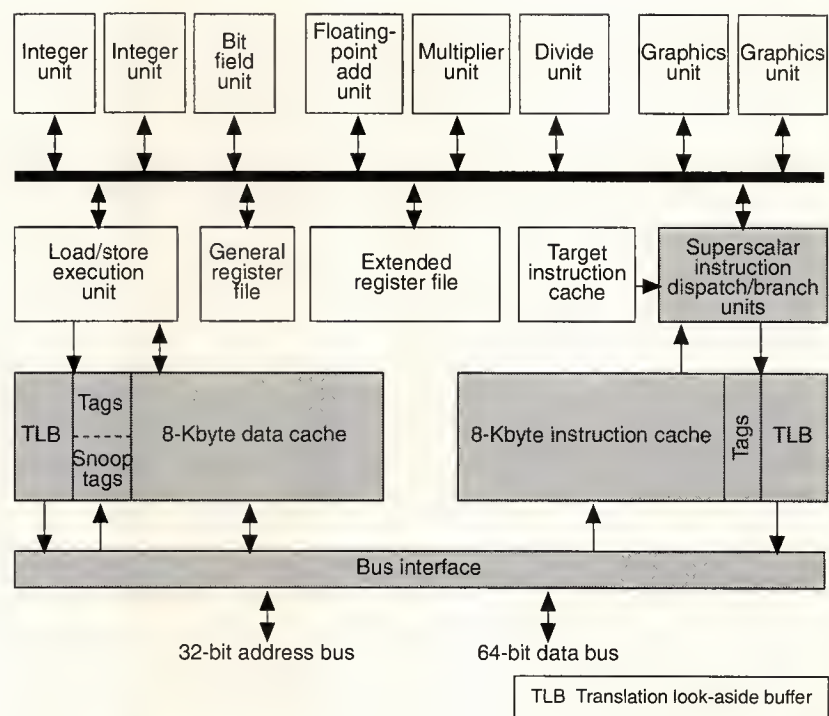


Figure 1. Block diagram of the 88110.

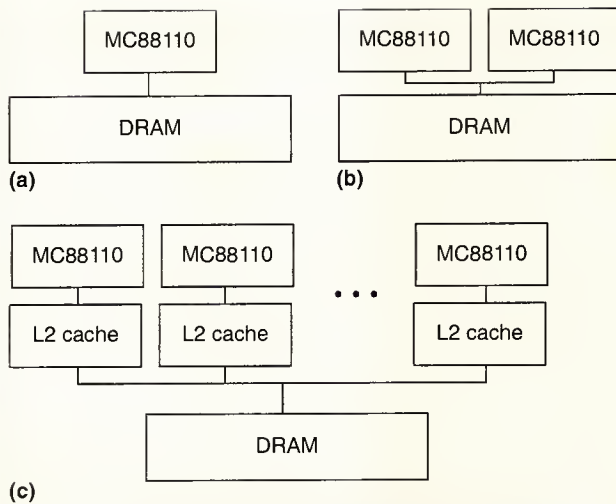


Figure 2. Target system configurations: single (a), dual (b), and multiprocessors (c).

memory environment. A common, 64-bit, external bus services cache misses. The demultiplexed, pipelined bus supports burst mode, split transactions, and bus snooping.

We designed the 88110 especially for the three basic system configurations shown in Figure 2:

- single processors tightly coupled to low-cost DRAMs;
- dual-processor systems, also coupled to inexpensive DRAMs, either in a symmetrical multiprocessing arrangement or with one of the 88110s dedicated to a particular function such as graphics or digital signal processing (DSP); and
- medium-scale shared-memory multiprocessor systems, with each processor using local secondary static RAM (SRAM) cache, which we call L2.

Instruction set architecture

To improve performance, we extended the instruction set architecture of the 88110 beyond that of the 88100 microprocessor. We enhanced a number of the integer and floating-point instruction sets and added a new set of capabilities to support 3D, color graphics image rendering. All the enhancements are upwardly compatible with the 88100; that is, the 88110 can run existing 88100 binaries.

Base architecture extensions. We made the following minor enhancements of the base instruction set:

- Extensions of integer multiply and divide to improve support for signed multiplication and for arithmetic on higher precision integers. Instructions permit multiplication of two 32-bit numbers returning a full 64-bit result, and division of a 64-bit number by a 32-bit number,

returning a 64-bit quotient.

- Additional information returned from the integer compare instruction to improve string-handling capability.
- Addition of static branch prediction to the branch opcodes, providing a mechanism by which the compiler gives the hardware a hint as to the direction a given conditional branch is likely to go. We estimate that the compiler potentially can statically predict more than 85 percent of the dynamically executed conditional branches correctly. We also believe that runtime branch profiling can further improve this rate in specific cases.
- An option that allows 16-bit immediate address offsets and literal constants to be treated as signed numbers (the 88100 treats them only as unsigned).

Floating-point architecture extensions. Our enhancements of the floating-point architecture were more significant. Anticipating heavy use of the processor for graphics and DSP and greater use of floating-point data in many general-purpose PC applications, we added the following:

- An extended floating-point register file to provide register name space for floating-point variables and frequently accessed constants beyond that provided in the 88100 architecture. The extended register file contains thirty-two 80-bit registers. Each register can hold one floating-point number of any precision—single, double, or double-extended. For compatibility with existing code, single- and double-precision floating-point numbers continue to be supported in the general register file as well. The compiler can use the additional register name space to improve code schedules and reduce memory references. This feature alone results in a speedup of more than 15 percent on the SPEC (Systems Performance Evaluation Cooperative) floating-point benchmarks.⁷ The speedup is substantially greater on many graphics and DSP-intensive routines. We expect further improvement as compilers learn to take better advantage of this feature.
- Hardware support for IEEE-754, 80-bit, double-extended precision data,⁸ to improve the accuracy and robustness of intermediate calculations in floating-point libraries.
- Hardware support for arithmetic on infinities, to eliminate the need for trapping to an IEEE software envelope to handle infinities, a frequent occurrence in some graphics algorithms.
- A time-critical floating-point mode to facilitate implementation of real-time DSP algorithms. In this mode, the hardware attempts to deliver an arithmetically sensible result rather than trapping on exceptional conditions such as underflow, overflow, and NAN (not a number).⁸ For example, the hardware flushes underflows to zero rather than trapping to generate the exact IEEE-specified denormalized result. This feature is useful in real-time

algorithms because it reduces execution time and eliminates data-dependent time variations, thereby increasing the amount of work that can be scheduled up to a given deadline.

Graphics architecture extension. Fast processing of 3D graphics viewing transforms and lighting calculations requires high floating-point performance. However, good floating-point performance alone is not sufficient for good graphics performance. Due to the large amounts of data involved, graphics images are usually represented and rendered in packed, low-precision, fixed-point formats. Conventional microprocessors are not well suited to processing these data types. The traditional solution of adding a special-purpose coprocessor to the system increases costs and creates the difficulty of seamlessly integrating another processor architecture into the software environment.

A new set of instructions gives the 88110 this graphics capability. The hardware to implement these instructions takes only a small incremental investment in silicon (approximately 2.5 percent), while substantially increasing performance on fixed-point shading and image processing. For many systems, these instructions eliminate the need for coprocessors or special-purpose external hardware. For systems demanding greater graphics performance, a dual-88110 system provides the coarse-grain parallelism of a coprocessor approach yet preserves a homogeneous programming and software environment.

The new graphics instructions accelerate operations on the fixed-point and integer data types (Figure 3a,b) found in many 3D, color image-rendering algorithms. They operate on these packed data types 64 bits at a time.

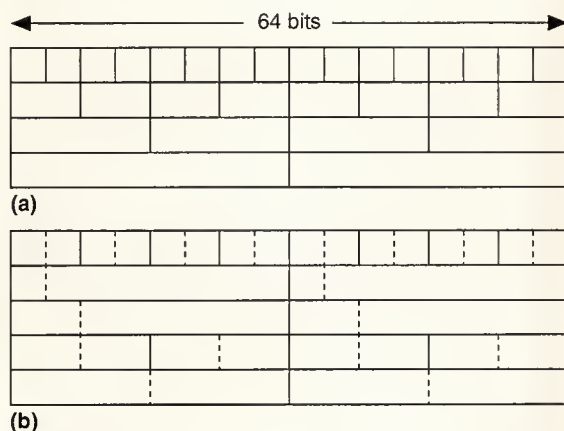


Figure 3. Graphics data formats: Packed integer data formats in pixels (a) and packed fixed-point data formats in color intensity values (b).

. padd.t	rD,rS1,rS2	; add fields
. padds.x.t	rD,rS1,rS2	; add fields with saturation
. psub.t	rD,rS1,rS2	; subtract fields
. psubs.x.t	rD,rS1,rS2	; sub fields with saturation
. punpk.t	rD,rS1	; pixel unpack
. ppack.r.t	rD,rS1,rS2	; pixel pack
. pmul	rD,rS1,rS2	; multiply
. prot	rD,rS1,rS2	; rotate
. prot	rD,rS1,<O5>	; rotate immediate
. pcmp	rD,rS1,rS2	; Z compare

Figure 4. Graphics instruction set.

The graphics instruction set (Figure 4) provides addition and subtraction on 8-, 16-, and 32-bit fields within 64-bit operands using either modulo or saturation arithmetic. Saturation arithmetic allows overflows or underflows within a field to clamp at the maximum or minimum value representable in the field rather than wrapping around as in normal modulo arithmetic. This method can be useful, for example, when addition of a color intensity value could result in an overflow that, in modulo arithmetic, would alias to a lower intensity value and thus produce an undesirable visual anomaly. Saturation is available on signed, unsigned, and mixed-sign numbers.

The set includes instructions for unpacking, truncating, packing, and rotating 4-, 8-, 16-, and 32-bit data fields to quickly convert between packed, fixed-point formats (intensity values) and packed, short, integer formats (pixels).

A graphics multiply instruction supports image-processing and -compositing algorithms, and a 64-bit compare instruction allows comparison of two pairs of 32-bit, fixed-point or floating-point Z-buffer values in one instruction.

Figure 5 (on the next page) is an example of how these primitive instructions can be chained together to implement complex image-processing operations such as compositing. A four-instruction sequence is illustrated. 1) The *punpk* instruction unpacks a 32-bit, four-channel, true-color pixel into four 16-bit, zero-padded, fixed-point numbers. 2) *Pmul* multiplies the result by an 8-bit integer, producing four new 16-bit, fixed-point numbers. 3) *Padd* adds these results to four other 16-bit, fixed-point numbers. 4) *Ppack* truncates the four 16-bit results of the addition to 8 bits, packs them together, and accumulates them with the pixel computed in the previous iteration of the loop by shifting the old pixel to the left and inserting the new pixel in its place. The program then can write this two-pixel result to the image buffer in memory, using a double-word store (St.d).

An important characteristic of the graphics instructions is their clean integration with the 88000 architecture, made possible by the 88000's special-function unit concept, which allows the instruction set architecture to be easily extended.

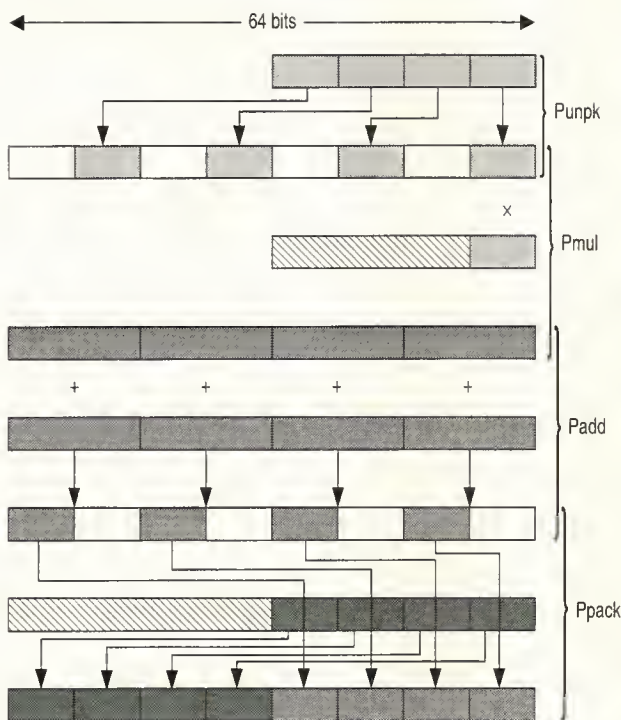


Figure 5. Graphics instruction chaining.

All the new instructions comply with the RISC philosophy of instruction set design.^{9,10} They all take two 64-bit operands from the general register file, perform a simple operation, and produce a single 64-bit result. No instruction side-effects or special-purpose "kludge registers" are introduced into the programming model. Since data is kept in the general register file, all the existing 88000 arithmetic, logical, and bit-field instructions can be freely applied to the graphics data. Furthermore, the storage of data in the general register file allows the fixed-point graphics operations to overlap floating-point graphics operations, creating a high-throughput graphics pipeline.

Instruction fetch and issue

The heart of the 88110 microarchitecture is a centralized instruction sequencer, which dispatches instructions into an array of parallel execution units (Figure 6). The sequencer fetches instructions from memory, tracks resource availability and interinstruction dependencies, directs operand flow between the register files and execution units, and dispatches instructions to the individual execution units.

On each clock cycle, the sequencer fetches two instructions from the instruction cache and two from the branch target instruction cache. It decodes the appropriate instruc-

tion pair while fetching the necessary data operands from the register files. If all the required execution units and operands are available, the sequencer simultaneously dispatches both instructions to their respective execution units.

Instructions leave the sequencer in strict program order. The sequencer always tries to dispatch two instructions; if it can't, it tries to dispatch at least the first of the pair. In that case, the second instruction moves into the first issue slot, a new instruction is fetched to replace it, and the new instruction pair tries to issue on the next clock cycle.

Although the sequencer always dispatches instructions in order, not all instructions *issue*, or begin execution, in order. Reservation stations^{11,12} in their respective execution units allow branches and stores to be dispatched even if their source operands are not available, so that further instruction dispatch can continue. Branch and store instructions wait in the reservation stations until the required source operands become available and the instructions can issue. Thus, branches and stores may issue out of order. This dynamic rescheduling¹³ ensures that branches and stores, which normally constitute about 30-40 percent of the dynamic instruction mix, rarely stall on data dependencies and do not delay the dispatch of subsequent instructions.

Once the sequencer has dispatched an instruction into an execution unit pipeline, the instruction proceeds at a pace set by the capability of that unit. When an execution unit finishes an instruction, the sequencer controls write-back of the results into the register file and forwarding of the results to any execution unit that needs them immediately. The sequencer ensures that no register conflicts exist, but it is otherwise free to update the register file out of program sequence. This out-of-order instruction completion model allows useful work to proceed under long-latency operations. It also al-

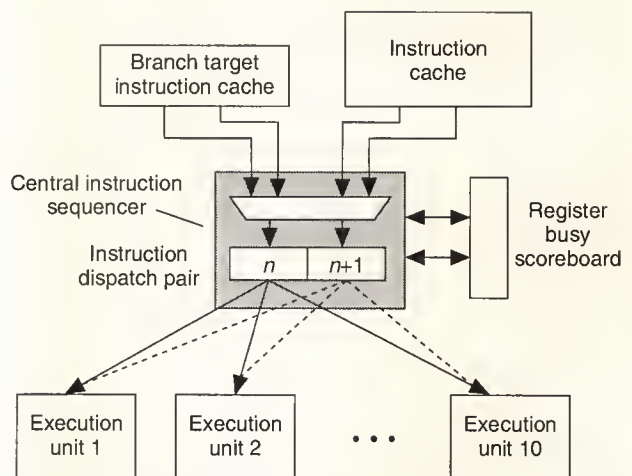


Figure 6. Instruction dispatch.

lows a mixture of execution unit types, possibly with variable-length pipelines, without resorting to a common, long, fixed-length pipeline with complicated pipeline bypass circuitry.

The master instruction pipeline, shown in Figure 7, is a conventional, four-stage RISC pipeline that completes most instructions in three clock cycles. In the first stage of the pipeline, the sequencer fetches an instruction pair from the instruction cache. In the second stage, it decodes these two instructions, fetches their operands from the register files, and decides whether or not to dispatch them into execution. Instructions execute during the third stage; for most instructions the execute stage requires one clock cycle, but some take more. In the fourth and final stage, the sequencer writes the results from the execution units into the register files.

Three things can prevent an instruction from issuing: 1) A necessary resource is not available or is busy (structural hazard); 2) an operand conflict exists with a prior instruction (data hazard); or 3) a branch causes a change in program flow, requiring an alternate instruction stream to be fetched,

thus temporarily starving the dispatch unit of instructions (control hazard).⁴

Structural hazards. Structural hazards occur because of pipeline resource or instruction class conflicts. Pipeline conflicts are rare in the 88110 because the register files are multiported with full-width data paths, and all execution units (except the divider) either execute in one cycle or are fully pipelined to accept a new instruction each clock cycle.

Instruction class conflicts occur when two instructions requiring the same execution unit attempt to issue on the same clock cycle; for example, two multiply instructions attempt to issue as a pair, but only one multiplier execution unit exists. The concurrency matrix in Figure 8 on the next page shows the relatively few pairings of instructions in the 88110 that will stall due to a class conflict. We eliminated a significant number of class conflicts by providing a duplicate set of integer ALUs (arithmetic logic units).

An important aspect of the 88110's superscalar instruction issue capability is that it is symmetrical; that is, any instruction can be dispatched from either slot in an instruction dis-

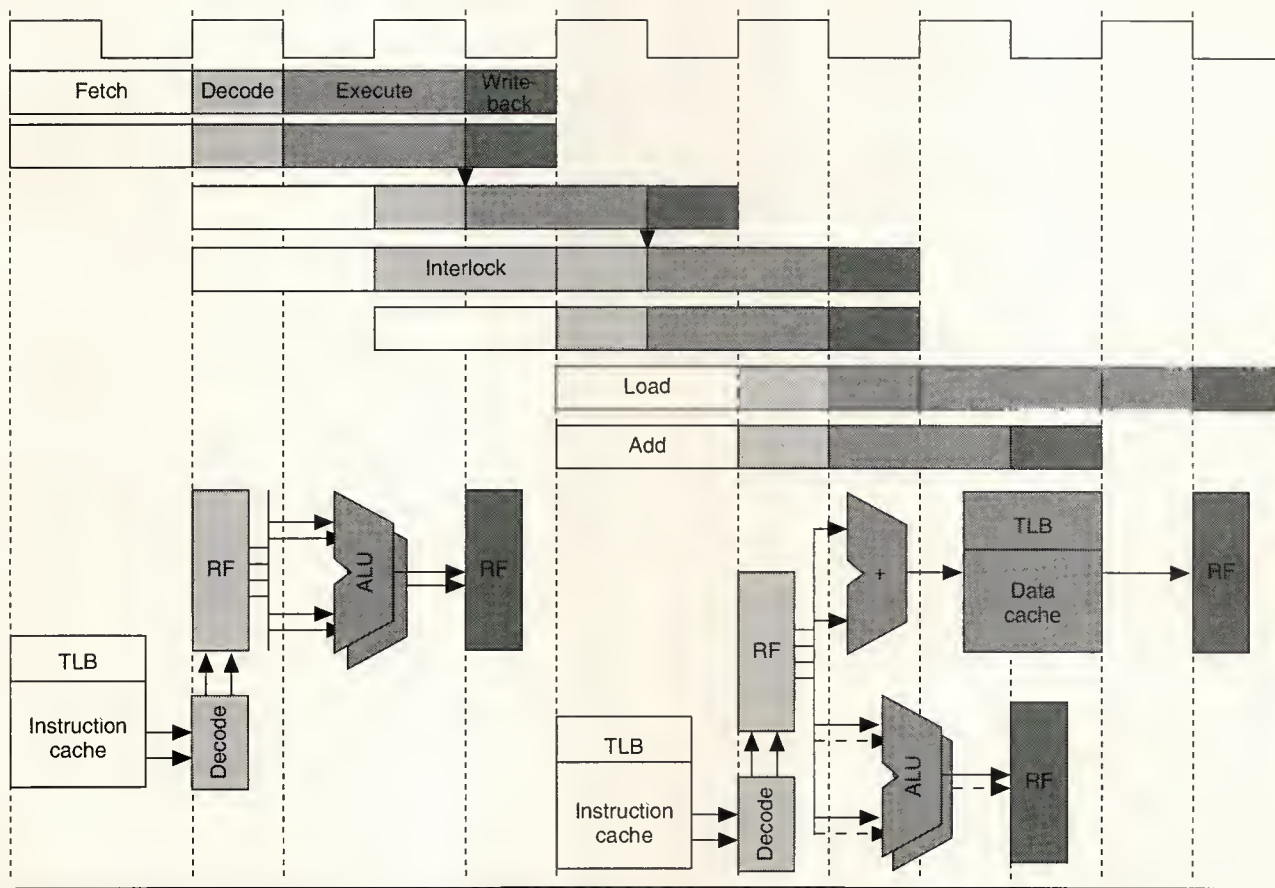


Figure 7. Master instruction pipeline; RF indicates register file.

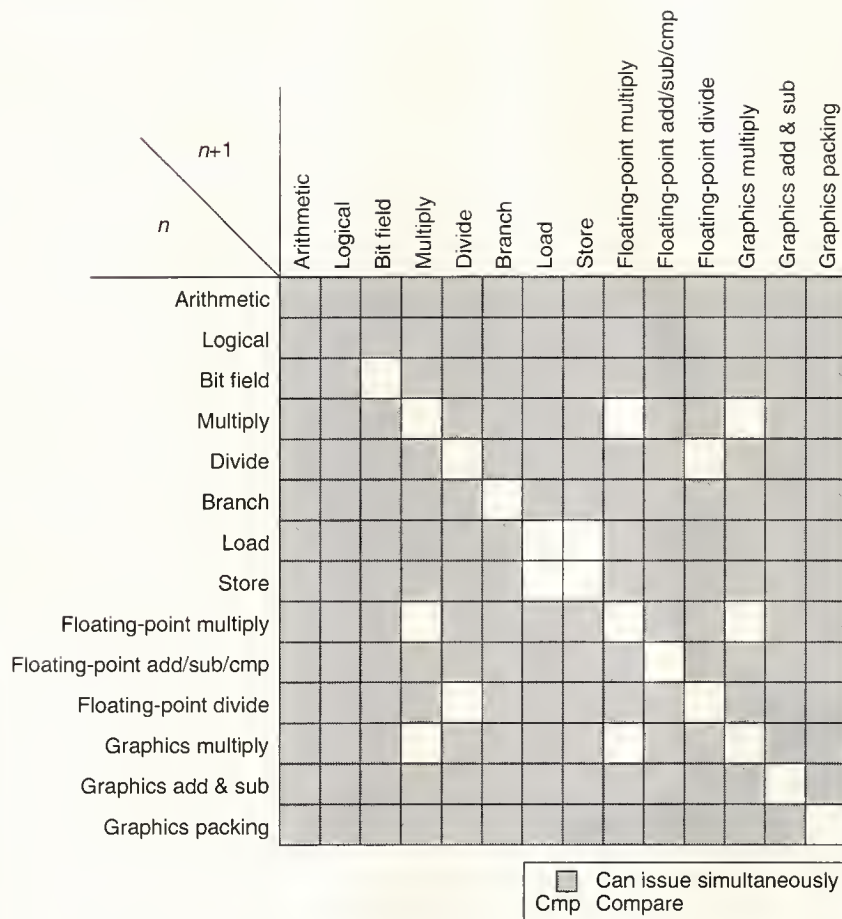


Figure 8. Instruction concurrency matrix.

patch pair, as illustrated in Figure 9. For example, the sequencer can dispatch a multiply instruction to the multiplier regardless of whether the instruction is in the first or second slot of a dispatch pair. Thus, the 88110 has none of the artificial instruction ordering or pairing restrictions characteristic of VLIW or restricted superscalar machines. Also, the sequencer fetches instructions from the instruction cache two at a time regardless of their address alignment, so no alignment restrictions must be met. Removing these constraints frees the compiler to optimize for more important considerations.

Data hazards. Instruction issue can also stall because of data hazards such as read-after-write (true data dependency), write-after-write (output dependency), or write-after-read (antidependency) hazards. Read-after-write hazards occur when an instruction needs a result from a previous instruction that has not yet completed execution. Write-after-write hazards occur when an instruction writes to a register after a

subsequent instruction has already written to the same register, thus leaving the register with old data. Write-after-read hazards occur if an instruction attempts to write a result to a register before a previous instruction reads the old value. The write-after-write and write-after-read hazards are really false dependencies, since they involve no true data dependency, only a register name conflict.

A register-busy scoreboard automatically interlocks the 88110 pipeline against incorrect data on hazards by tracking source and destination operand availability. Each time the sequencer dispatches an instruction, it also marks the instruction's destination register as busy until the instruction completes execution. As the sequencer considers instructions for dispatch, it checks the scoreboard to ensure that no register conflicts exist with prior instructions still in execution. (The term *scoreboard*, as applied to computers, originally referred to the complex centralized queue and reservation mechanism used in the CDC 6600 for tracking all aspects of out-of-order execution.¹⁴ Recently, however, the term has become generic, referring to any control unit that handles register reservations¹²—including much less sophisticated units than the CDC 6600's—such as that in the 88110.)

The sequencer avoids incorrect data on read-after-write hazards by checking the source operand register scoreboard

bits and on write-after-write and on write-after-read hazards by checking the destination operand register scoreboard. The sequencer can dispatch branches and stores even if their source operand is busy. However, they are held in a reservation station and do not begin execution until the scoreboard

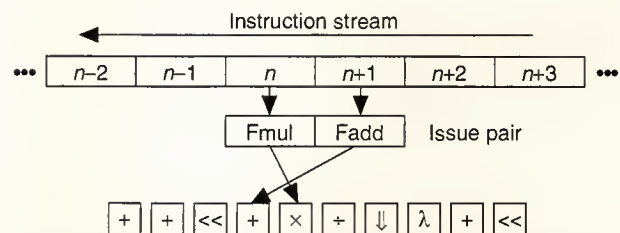


Figure 9. Symmetrical superscalar instruction issue.

bit for the needed register clears and the operand can be read.

Of the three types of data hazards, read-after-write hazards cause the most instruction issue stalls in the 88110. We keep these stalls short by providing 1) low-latency execution units and 2) a register file bypass from the execution unit result buses to the execution unit inputs. The bypass makes instruction results available immediately to subsequent instructions without having to wait an extra clock cycle for results to be written into and read out of the register file.

For the most part, the 88110 relies on static scheduling of the instruction stream to avoid stalling on hazards. In many cases, static scheduling is straightforward and the compiler can handle it effectively. However, statically scheduling code around some types of data hazards can be difficult or inefficient; that is why the 88110 performs dynamic rescheduling of branches and stores.

As an example of dynamic rescheduling of stores, consider the common operation of fetching data from memory, performing some computation on it, and then storing the result back in memory. If the computation requires multiple cycles—as a floating-point multiply does, for example—the store of the result introduces a data hazard that would stall instruction issue even though no further need exists for that data in the program. The store reservation stations allow stores to be set aside and instruction issue to continue while the store data is being computed. Then, when the store data becomes available, the sequencer immediately forwards it to the appropriate reservation station and allows execution of the store to begin.

Similar stalls could occur on conditional branch operand data hazards—due either to long-latency operations (such as load-branch sequences) or to dispatch pair dependencies (such as compare-branch pairs). As with stores, the 88110 provides a reservation station to avoid stalling on these branches. In the case of branches however, an additional problem exists. The machine does not know where to continue execution until the branch operand is available and the sequencer can evaluate the condition. Therefore, the sequencer predicts the branch direction, and instructions down the predicted path execute conditionally, or speculatively, until the branch operand is resolved. The static prediction of the branch direction is based on the opcode of the branch instruction.

The branch reservation station provides

a place to set aside the branch instruction so that instruction issue can continue while the branch condition is being resolved. Once the operand becomes available and the condition is evaluated, the machine determines whether or not instruction execution actually went down the correct path. If it did, useful work was accomplished and execution simply continues uninterrupted. If the prediction was incorrect and execution went the wrong way, the machine backs up to the branch, undoing all changes made to the registers by conditionally executed instructions, and resumes execution down the other path.

Figure 10 contrasts the pipeline situation with and without speculative execution on a taken conditional branch (bcnd) that is dependent on a load. With speculative execution and branch prediction (top), the new instruction stream (target 0, target 1, and so on) begins execution immediately with no bubbles introduced into the pipeline. (By *bubbles* we mean lost opportunities for instructions to issue.) Without speculative execution and branch prediction (bottom), the machine would continue fetching down the sequential instruction stream (next 0, next 1), since the target address would not yet be available. Also, instruction dispatch down the target

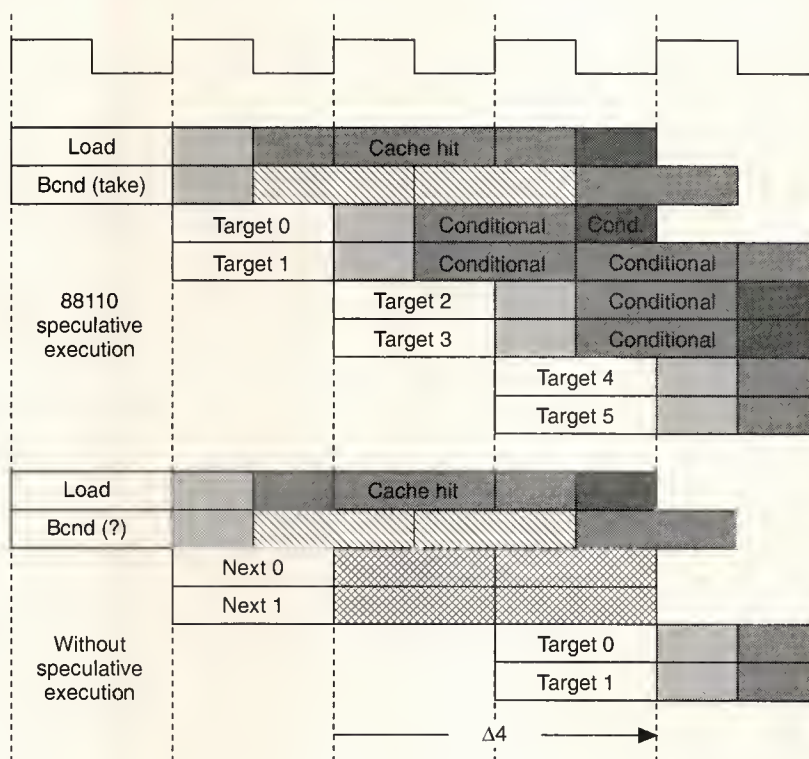


Figure 10. Speculative execution.

path would have to delay for two clock cycles (four bubbles) while the machine waits for load data with which to compute the branch direction.

During speculative execution, the instruction fetches that miss the instruction cache access the bus just as in normal execution. Load instructions can access the data cache on a hit, but the bus will not service a data cache miss until the branch condition resolves. Store instructions can be dispatched to the reservation stations but can never access the cache or the bus during conditional execution. This procedure prevents corruption of the memory image by a store instruction that is eventually canceled on a misprediction.

The accuracy of branch prediction and the penalty for mispredicting are important, of course, to the overall performance gain realized from speculative execution. Although prediction accuracy depends on the compiler being used and the nature of the application, our simulations indicate that good static-prediction accuracy is achievable. On the SPEC benchmarks over 80 percent of all conditional branches take the anticipated path, and over 70 percent of the branches that need to be predicted are being predicted correctly. Currently, our compiler predicts only the simple branch cases, so we expect these results to improve as the compiler becomes more aggressive. Also, we are currently seeing a penalty of less than one-half percent for mispredicting branches, although the penalty may increase on applications that allow deeper speculative execution.

Control hazards. Generally speaking, when a pipelined processor encounters a branch, it needs time to evaluate the condition, compute the target branch address, fetch instructions from the new target instruction stream, and refill the pipeline. The 88110 deals with pipeline bubbles caused by control hazards by means of the speculative execution model just described and by the use of a branch target buffer to shorten branch execution latency.¹⁵

The speculative execution model permits out-of-order instruction execution to extend beyond the domain of a basic block. It also helps keep the instruction pipeline full and the execution units busy even in the face of small basic blocks, but it does not address branch latency.

RISC designers traditionally compensated for branch latency by using a branch-and-execute⁹ or delayed-branch¹⁶ strategy to give the processor something to do while the branch executes. In a superscalar design, however, a single architectural delay slot is insufficient to cover the two instruction bubbles inserted into the instruction pipeline by each clock cycle of branch latency. Short branch latency is important, even with speculative execution, to minimize the number of instructions subject to cancellation in the event of misprediction.

Due to the critical importance of control hazards to performance, we invested a significant amount of circuitry in the 88110 to reduce branch latency. During the instruction de-

code phase of the pipeline, the sequencer fetches two instructions at the branch target address from the branch target instruction cache (TIC) and supplies them as the first two instructions down the branch-taken path. The sequencer evaluates (or predicts, if necessary) the branch condition early in the pipeline to select either the target instruction pair from the TIC or the next sequential instruction pair from the instruction cache in time for the next instruction decode phase. By this time, with the branch target address computed, the instruction cache can supply further instructions along the target instruction stream. Thus, on a hit, the TIC can fill the two branch pipeline delay slots with useful instructions.

The TIC has 32 entries and is fully associative. Each entry in the TIC holds the first two instructions from a recently taken branch target path. The hardware automatically loads

***Even with its heavily pipelined,
out-of-order execution model,
the 88110 implements fully
precise exceptions.***

cache entries on a miss, following a FIFO replacement policy. The sequencer uses the logical address of the branch being evaluated to index the TIC. Thus, the target instructions are available immediately for the next instruction fetch phase of the pipeline.

TIC hit rates depend heavily on the application, but our simulations show an average TIC hit rate of around 85 percent on the SPEC benchmark suite.

Exceptions

Program exceptions and interrupts have always been a problem in pipelined machines, especially those that execute instructions out of order and/or speculatively. During normal program execution, machine state changes can occur out of program sequence as long as the state currently relevant to the program *appears* to be in the correct program order. But in a parallel machine, the internal pipelines and buffers temporarily hold much of the dynamic machine state. Thus, at any point in time, the register files do not completely reflect the true current machine state. So, when a program exception occurs and the instantaneous state of all the registers become manifest, the dynamically held state can be lost or confused. The register file's inconsistency with the actual machine state makes correct recovery from the exception difficult or impossible.

Even with its heavily pipelined, out-of-order execution

model, the 88110 implements fully precise exceptions. That is, the processor always presents the architecturally correct state to an exception-handling routine. It also gives an exact indication of which instruction caused the fault and where to resume execution. The 88110's precise exception model dramatically simplifies and speeds up exception- and interrupt-handling software routines.

When an instruction generates an exception—for example, a page fault or an arithmetic overflow—instruction execution continues until all instructions that issued prior to the faulting instruction complete. (This step ensures that all synchronous exceptions occur in strict program order). At this point, execution stops, the internal pipelines are cleaned up, and the machine backs up to the instruction that caused the exception, leaving all registers in the precise architectural state that existed before the faulting instruction issued. The 88110 accomplishes this by means of a history buffer,¹⁷ which records the relevant, user-visible machine state as instructions issue. The processor uses information stored in the history buffer to quickly restore the machine state back to the point of the exception. This is the same mechanism the 88110 uses to recover from mispredicted branches.

When the machine recognizes an asynchronous external interrupt, it halts execution, aborts all unfinished instructions (or waives write-back in the case of a memory transaction in progress on the bus), and backs out the effects of any instructions that completed out of order. This procedure mini-

mizes interrupt response latency, a critical parameter in many real-time system applications.

Register files

The 88110 has two sets of register files, the formats of which are shown in Figure 11a,b. The general register file primarily holds fixed-point values and address pointers; the extended register file holds floating-point data.

The general register file has thirty-two 32-bit registers, which can be used by all instructions in the machine. These registers are accessible in pairs to supply 64-bit operands whenever necessary—for example, for graphics and double-precision floating-point values. Register zero is hardwired to the integer constant zero (0).

The extended register file is a new addition to the original 88000 architecture. It contains thirty-two 80-bit registers, which are used exclusively by the floating-point instructions. Each extended register can hold one floating-point number in either single, double, or double-extended format. Register zero in the extended register file is hardwired to the floating-point constant positive zero (+0.0E00). We provided instructions that quickly move data back and forth between the two register files.

Both eight-ported register files can supply all the operand bandwidth required to sustain the peak instruction issue rate of two instructions each clock cycle, regardless of the instruction mix or data precision. Data-forwarding paths around

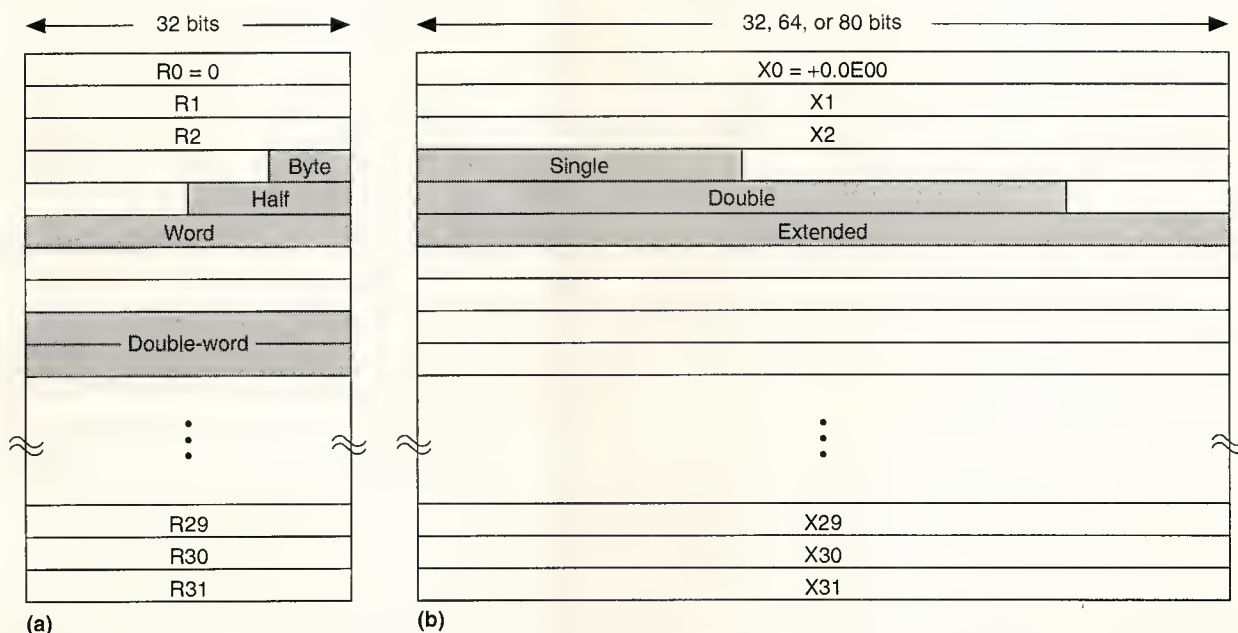


Figure 11. Register files: General (a) and extended or floating-point (b).

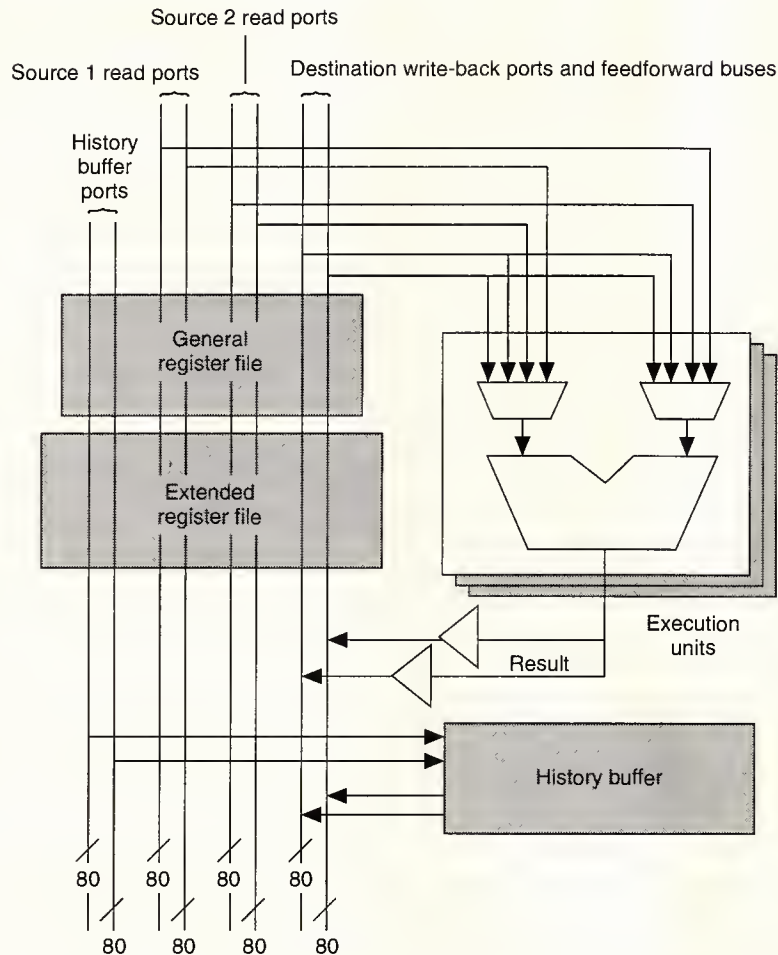


Figure 12. Operand data paths.

the register files route a result returning from an execution unit directly to the inputs of a waiting execution unit while the result is also being written into the register file (see Figure 12). This approach avoids stalling the instruction issue an extra clock cycle while data is written into the register before it can be read out on a source port.

Execution units

The 88110 contains 10 independent execution units: branch, integer (two), bit field, multiplier, floating-point adder, divider, graphics (two), and data or load/store. The dataflow paths from outside the chip, through the caches and register files, and into and out of the execution units, are shown schematically in Figure 13.

Integer units. The integer units are simple 32-bit ALUs that handle all the fixed-point arithmetic instructions and logical instructions. The execution latency of both integer units

is one clock cycle.

Bit field unit. This unit is a shifter/masker circuit that handles the 88000's extensive set of bit-field manipulation instructions. It also has a single-clock-cycle execution latency.

Multiplier unit. The multiplier unit handles all 32- and 64-bit signed and unsigned integer multiplies, the graphics multiply, and the single-, double-, and extended-precision floating-point multiplies. The fully pipelined unit can start a new multiply instruction every clock cycle. The multiplier has an execution latency of three clock cycles for all data types. The 32×64 -bit multiplier uses Booth partial product generators and a Wallace tree to sum the partial products twice each clock cycle to maximize circuit efficiency.

Floating-point adder unit. The floating-point adder executes all single-, double-, and extended-precision floating-point add, subtract, compare, and integer conversion instructions. The fully pipelined unit can start a new instruction on every clock cycle. The adder has a three-clock-cycle execution latency for all precisions. A special shortcut reduces the latency for floating-point compare to one clock cycle. The dynamic 64-bit adder circuit uses a combined block-carry-look-ahead and fast-carry-select scheme. The actual 64-bit add time is much shorter than one clock cycle. Most of the three-clock-cycle execution time occurs because of the floating-point format operations such as reserved-oper-and check, exponent debiasing, mantissa

alignment, normalization, and rounding.

The construction of a fully pipelined floating-point multiplier and adder with short latencies is very hardware intensive. The return on this investment is the ability to achieve a more efficient static code schedule and an extremely high floating-point throughput. Long-latency operations require that a large number of independent instructions be available to be scheduled into the pipeline delay slots to avoid bubbles and keep execution unit usage high. In general, the compiler needs to find less program parallelism on hardware with short latencies than it does on hardware with long latencies to achieve the same level of performance.

One commonly used technique for scheduling around long-latency operations is loop unrolling. This technique increases the basic block size and the number of data-independent operations available to be scheduled into pipeline delay slots. A difficulty with loop unrolling is that it requires a large reg-

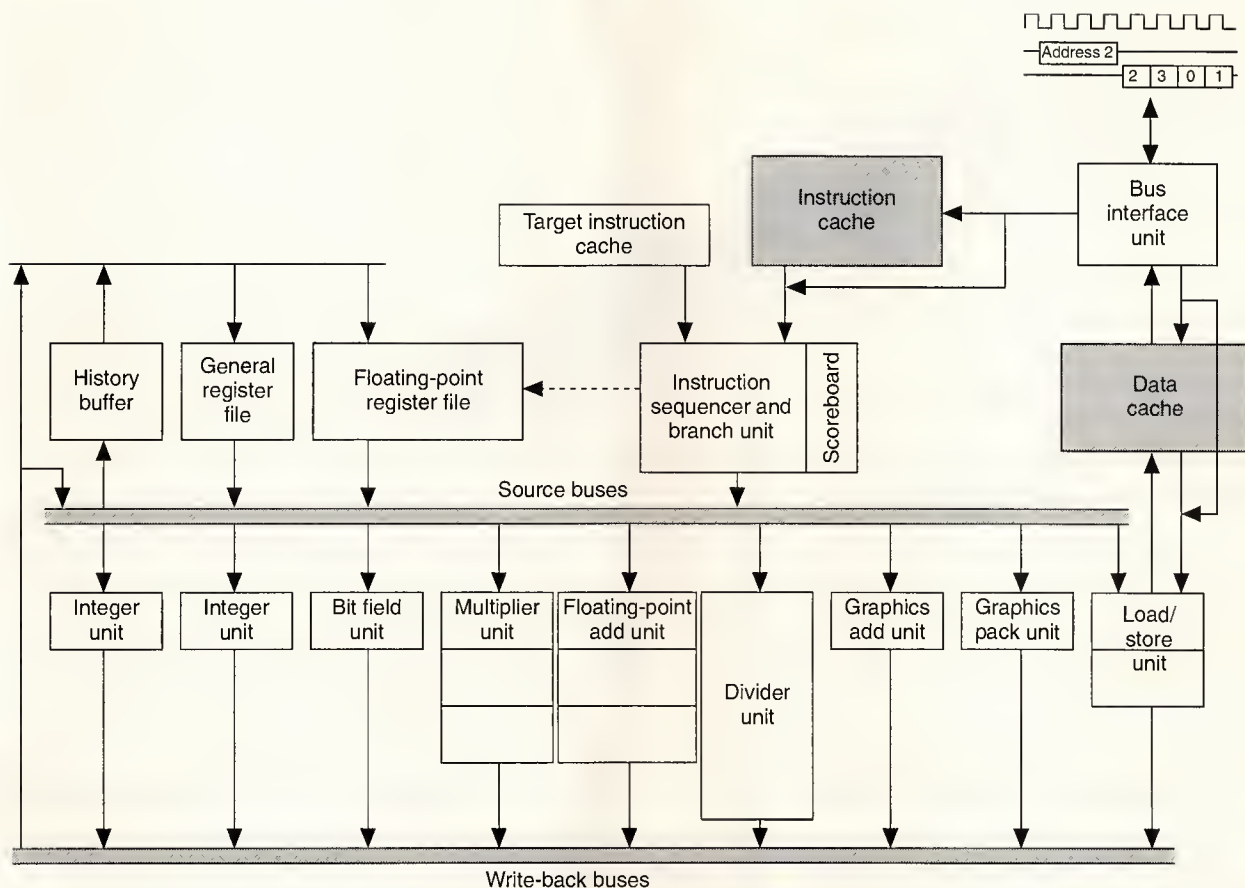


Figure 13. Organization of the 88110.

ister name space so registers can be allocated to avoid data hazards. It also increases the static code size, which can waste memory space and instruction cache entries. The 88110's short, three-cycle floating-point latency is well balanced with the large register files, making a small amount of loop unrolling effective without requiring elaborate register-renaming hardware.¹¹

Divider unit. The divider handles all 32- and 64-bit signed and unsigned integer divides and all single-, double-, and extended-precision floating-point divides. The iterative divider uses a radix-8-per-clock SRT algorithm (Sweeney-Robertson-Tosher) with a latency dependent on the operand type and precision. The execution latency of single-precision floating-point division equals 13 clock cycles.

Graphics units. Two execution units implement the new graphics instructions. One handles the arithmetic operations, and the other handles the bit-field packing and unpacking instructions. Both units have a single-clock-cycle execution latency. Because the two units are independent, each can accept a new instruction every clock cycle. In fact, the in-

structions are partitioned in a manner that often makes it possible to schedule graphics algorithms to sustain a throughput of a full two instructions each clock cycle. As an example, Figure 14 on the next page shows execution of the inner loop of a simple Gouraud shading algorithm. Since the graphics units behave the same as other execution units, graphics instructions can issue together with any other integer, floating-point, or memory-referencing instruction. This flexibility minimizes loop overhead and allows very efficiently scheduled graphics routines.

Load/store unit. The load/store unit is the most sophisticated execution unit in the 88110. We invested a considerable amount of circuitry in this unit because of the critical importance of memory referencing to overall performance. The unit provides a *stunt box*¹⁴ capability for holding memory references that are waiting for the memory system and allows dynamic reordering of loads past stalled stores. (The CDC 6600's stunt box did not allow loads to pass stores, but the term *stunt box* has come to refer to any device that allows reordering of memory references in the memory system.¹²)

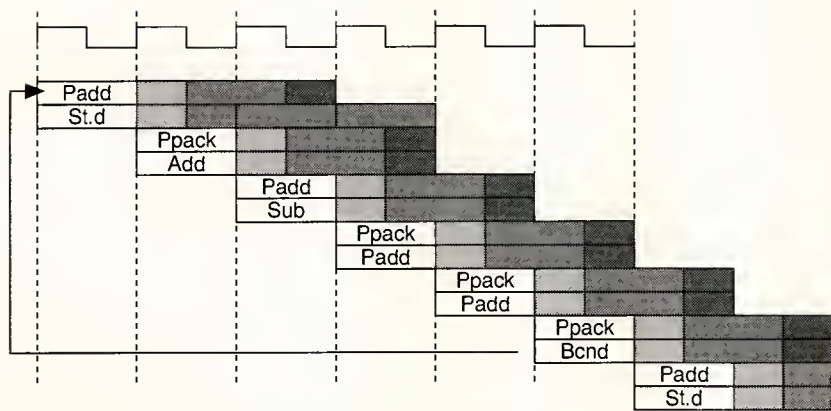


Figure 14. Graphics execution unit parallelism.

The load/store unit executes all instructions that transfer data between the data cache, or bus interface, and the register files. The data path from the load/store unit to the register files is a full 80-bits wide. Load latency for 32- and 64-bit data on a cache hit is two clock cycles—one longer than a normal integer add instruction.

On each clock cycle, the unit can accept one new load or store instruction from the instruction dispatch unit. When an instruction is dispatched to the load/store unit, it awaits access to the data cache in either the load queue or the store queue¹² (see Figure 15). Normal instruction dispatch and execution can continue while these instructions await service by the cache or memory system. On properly scheduled code, this buffering provides considerable tolerance of long memory latency.

The load queue is a simple, four-deep FIFO queue. The store queue is a somewhat more complex three-deep reservation station that is also managed as a FIFO queue. Since store instructions can be dispatched before the store data operand is available, store instructions wait in the store queue until the instruction computing the required data completes execution. When the operand becomes available, the sequencer directs it into the store reservation station, and the associated store instruction becomes a candidate for access to the data cache.

If a store instruction stalls in the reservation station waiting for its operand, subsequently issued load instructions can bypass the store and immediately access the cache. An address comparator detects address hazards and prevents loads from going ahead of stores to the same address, thus getting stale data. This load/store reordering feature allows runtime overlapping of tight loops by permitting loads at the top of a loop to proceed without having to wait for the completion of stores from the bottom of the previous iteration of the loop.

The data cache is nonblocking, or lock-up free,¹⁸ for store-

load accesses. For example, when a load bypasses a store and misses the cache, the cache can be decoupled from the bus so that the store can access the cache while the bus waits for memory. This is also true for a store miss followed by a load hit and for the user-mode touch-load instruction.

Touch-load provides a limited form of decoupling of load-store and load-load sequences. This instruction provides a mechanism for a program to bring a cache line into the cache before it is actually needed. While the load/store unit waits for memory to deliver the data, instruction issue can continue unrestricted. During that time, load and store instructions can

access the cache. The programmer can use the touch instruction to prefetch data into the cache to avoid load misses that would likely stall execution if serviced on demand. When used properly, these instructions can significantly increase cache hit rates and minimize load miss penalties.

The load/store unit implements other user-mode instructions to allow more effective scheduling of the data cache. An allocate instruction allocates a line in the cache without first bringing the line from memory. A program can use this instruction to avoid unnecessary bus transactions in cases where it will overwrite the entire cache line anyway. In addition, a line flush instruction can force a cache line out to memory. The line flush provides a mechanism to update a video frame buffer without allocating the frame buffer as write-through storage and thereby sacrificing the burst-mode line transfer capability of the bus.

The load/store unit also contains a selective cache bypass¹⁹ capability for stores. Store instructions can be selectively marked (with an opcode bit) to "store-through" the cache. Such a store bypasses the cache and proceeds directly to memory. If the reference hits the cache, the cache is updated; but if it misses the cache, no new line is allocated. This feature prevents pollution of the cache with data known to be of no further use to the program. It can increase cache hit rates by avoiding replacement of more useful entries in the cache. It can also improve cache miss latency by reducing the number of "dirty" lines that have to be copied back to memory when a new cache line is allocated.

The load/store unit also executes the 88000's XMEM instruction, which performs an atomic read-write operation that exchanges the contents of a register with a memory location. A program can use this instruction to implement semaphores and shared-resource locks in multiprocessor systems.²⁰ It can be used as a primitive to construct a wide variety of complex synchronization protocols, such as spin-lock, compare-and-

swap, and fetch-and-add. For example, one can construct an efficient spin-lock by repeatedly polling a lock with loads, which will hit in the cache and therefore not generate bus traffic. When the current owner of the lock releases it, the cache coherence logic brings the new copy of the lock into the cache, and the processor can then try to acquire the lock with an XMEM. The resulting indivisible read-write bus transaction ensures exclusive ownership of the lock.

The 88000 architecture uses primarily a "big-endian" byte order—that is, an address points to the most significant byte of a datum in memory—as opposed to a "little-endian" order—in which an address points to the least significant byte of a datum. The 88110 provides a solution for heterogeneous big/little-endian multiprocessor systems with a mode switch that allows data memory references to be performed in either big- or little-endian fashion. In little-endian mode, the load/store unit swaps the bytes in all half-words, words, double-words, and quad-words as they transfer into or out of the cache.

Address translation facilities

The 88110 offers full hardware support for a demand-paged virtual memory system.²¹ It provides hardware facilities for translating logical effective program addresses to physical memory addresses, for protecting areas of memory from unprivileged accesses, and for trapping to supervisory routines on accesses to pages not currently in memory. The organization of the address translation facilities is diagrammed in Figure 16.

The processor contains two independent and concurrent translation look-aside buffers—one for translating instruction addresses, the other for data addresses. Each fully associative TLB can hold thirty-two 4-Kbyte page address translation entries. Each entry contains a translation descriptor that maps a virtual page to its corresponding physical page number.

On each memory reference, the hardware looks up the logical address (which is equal to the virtual address in the 88110) by simultaneously comparing it to all en-

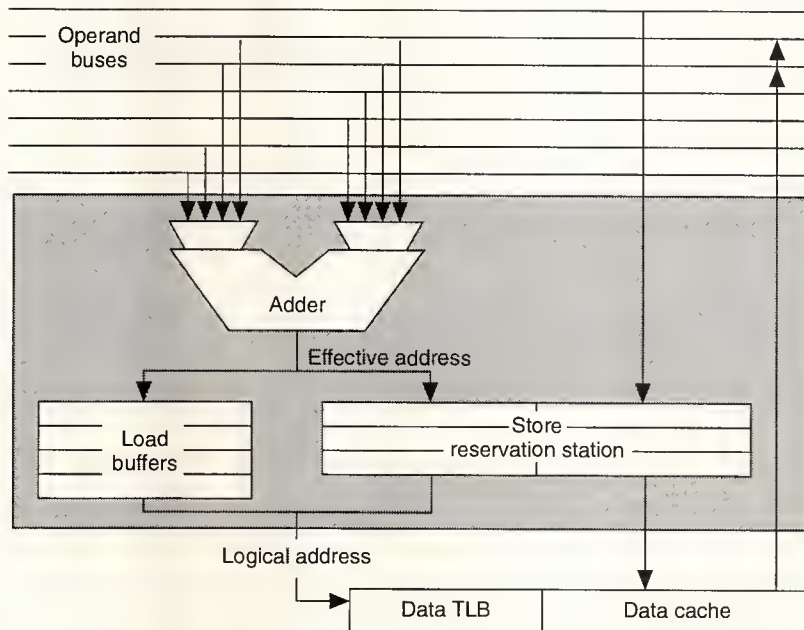


Figure 15. Load/store execution unit.

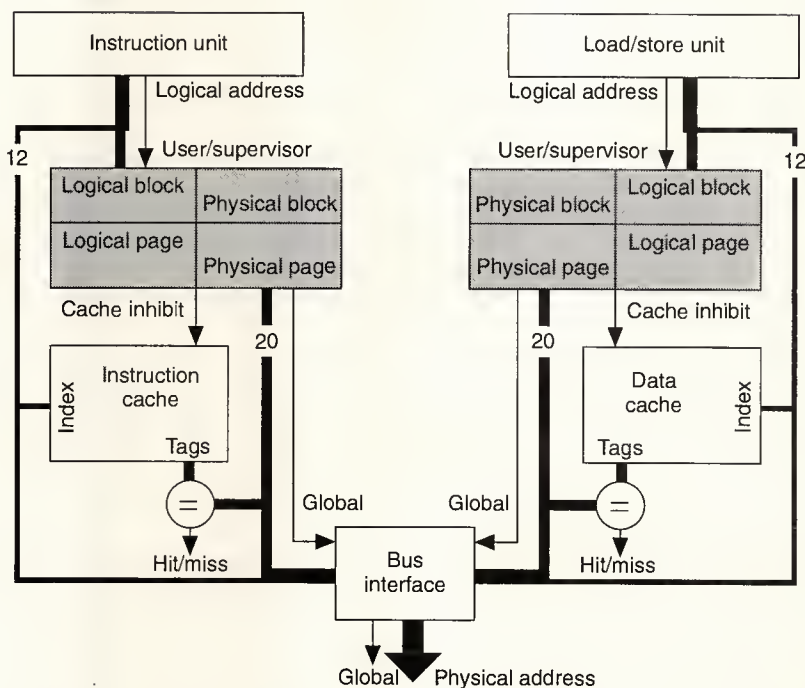


Figure 16. Virtual address translation facilities.

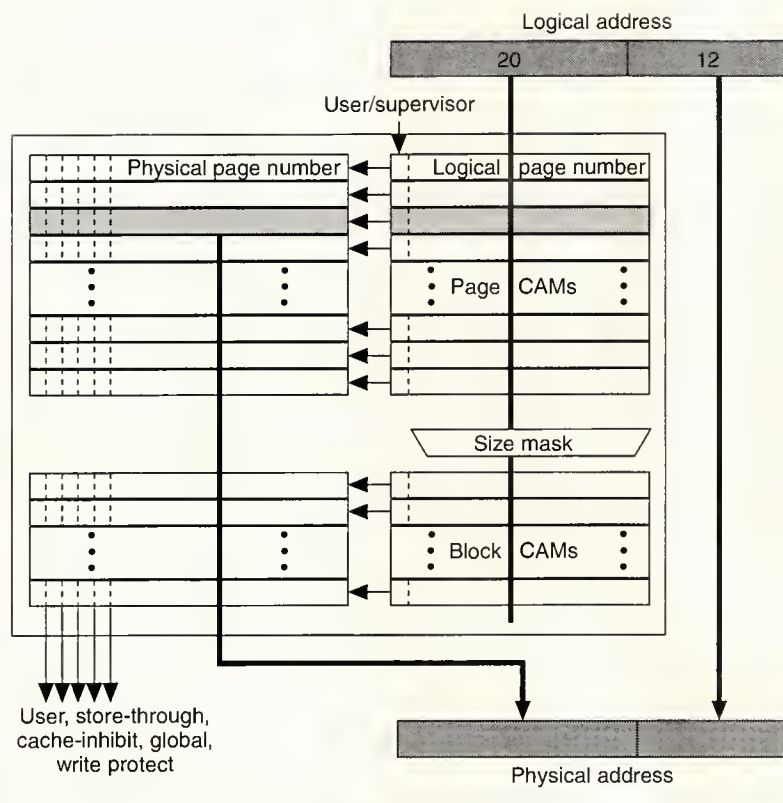


Figure 17. Translation look-aside buffer (TLB).

tries in the TLB, using content-addressable memory (CAM) elements as illustrated in Figure 17. Each CAM element is associated with one physical page descriptor, which is loaded into the TLB from the memory-based page tables maintained by the virtual memory operating system software.

If the TLB lookup finds an entry that matches the logical address of the memory reference being translated, it is a TLB hit and that entry is used to translate the address. If the memory reference does not violate the access privileges specified by the selected TLB entry, the page offset bits from the least significant 12 bits of the logical address form the final translated physical address. If the reference does violate the access privileges, the processor aborts the memory reference and signals an attempted memory protection violation to the operating system by taking an access exception trap.

Information stored in the TLB also governs certain caching policies, such as global access and cache bypass. The global property indicates that the referenced memory page is shared, and, therefore, other processors on the bus must "snoop" (watch for) any external bus transaction generated as a result of a cache miss to this page. The cache-inhibit and write-through properties allow data cache bypass to be controlled on an address basis at the granularity of a page. When a page

is marked "cache-inhibited," all references—loads or stores to that page—bypass the cache. If a page is marked "write-through," all stores to that page bypass the cache. This write-through capability is similar to the selective cache bypass (store-through) feature described earlier, but it allows bypassing on an address (page) basis rather than an instruction basis.

If the memory reference matches an entry in the TLB (a hit) but the entry is marked "invalid," the hardware generates a page fault trap, and control transfers to a supervisory routine. This routine would bring the accessed page in from disk, update the system page tables, and reissue the faulting memory instruction.

If the memory reference does not match any entry in the TLB (a miss), one of two things can happen. The hardware can automatically walk through the operating system's page tables to load a new descriptor into the TLB before starting the memory transaction. Or, the hardware can generate an exception, invoking a software routine to manually load a new descriptor into the TLB and then rerun the faulting memory instruction. The software TLB-reloading mechanism provides

the flexibility to support virtual memory management systems that use table structures (such as inverted page tables) different from those supported directly by hardware.

The simple but efficient hardware table-walking algorithm illustrated in Figure 18 indexes through two levels of system segment and page tables to locate a page descriptor, which is then brought into the TLB. The algorithm includes an indirection capability, which treats the resolved page descriptor as a pointer that is followed one additional level to a final page descriptor. Indirection allows multiple virtual addresses that map (alias) to the same physical address to be mapped through a common page descriptor. This capability simplifies system maintenance of the page referenced and modified status bits, typically used to implement an efficient demand-paged, virtual memory management system.

Each TLB implements facilities for the operating system to determine whether a particular translation is currently in the TLB, for locating and invalidating individual entries in the TLB, and for invalidating all user or supervisor entries. These facilities support many aspects of virtual memory management, including TLB coherence protocols such as the Mach operating system's TLB "shoot-down" algorithm.²²

Another important feature of the TLBs is the block address

translation facilities. In addition to the 32 page entries already described, each TLB has eight variable-size block address translation entries. The block translation entries perform address translation in a similar fashion to the page entries but are capable of mapping large blocks of memory (512 Kbytes to 64 Mbytes). These entries allow mapping of large static areas of system code or data structures and large entities such as frame buffers, without using an excessive number of TLB page entries.

Caches

The 88110 has a Harvard-style internal architecture—that is, it has separate, independent instruction and data paths. An on-chip instruction cache feeds the instruction unit, and an on-chip data cache feeds the load/store execution unit. Cache misses are multiplexed together and are serviced from a common external bus interface.

The instruction and data caches are both physically addressed. Physical caches have the advantage over logical caches in that synonyms do not occur. As a result, special precautions to disambiguate logical addresses across different process contexts are not necessary. No extra hardware is required for associating a logical address with a specific process, nor is it necessary to incur the overhead of flushing the caches on a context switch. Physically addressed caches also simplify maintenance of cache coherency in multiprocessor systems.

In the 88110 implementation, the caches are logically indexed and physically tagged, as illustrated in Figure 19. The cache arrays are directly indexed with the 12 untranslated page offset bits from the least significant portion of the logical address. Each cache line is tagged with the high-order 20 bits of the fully translated physical address. This arrangement allows selection of the cache set and retrieval of the cache tags and data in parallel with the translation of the logical address to a physical address in the TLB. After the physical address becomes

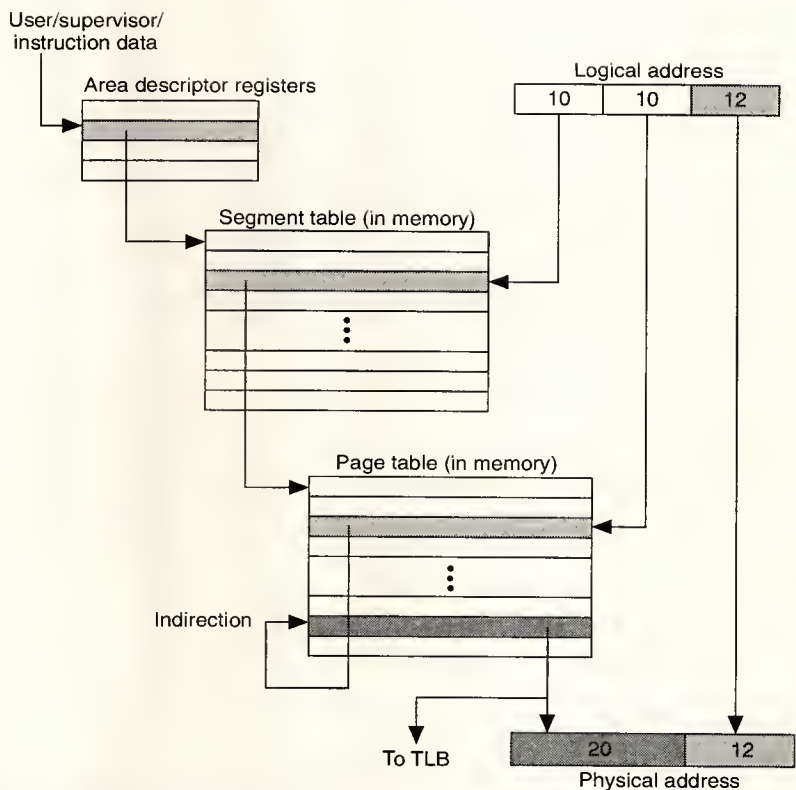


Figure 18. Automatic hardware table-walking scheme.

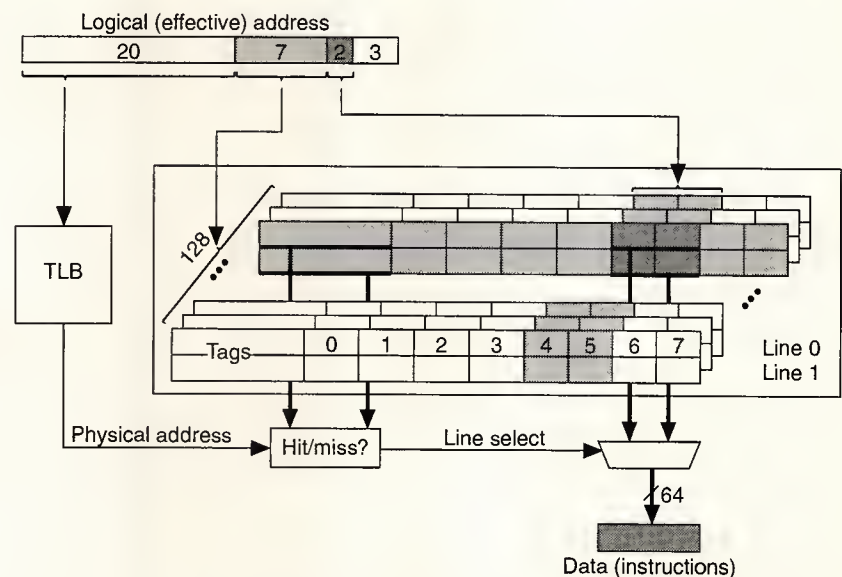


Figure 19. Organization of instruction and data caches.

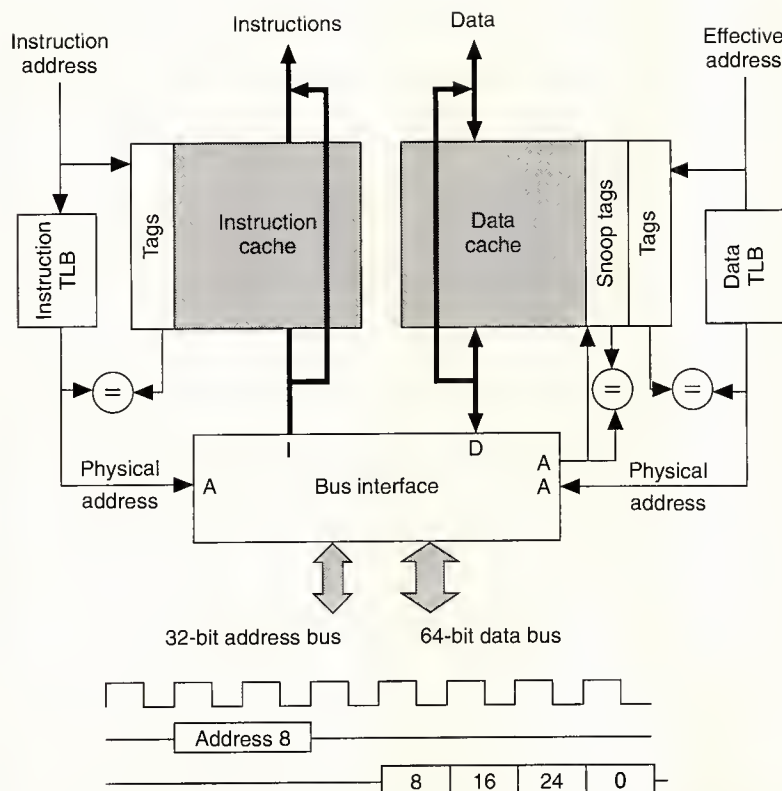


Figure 20. Cache miss.

available, the machine compares it against the two tags (one associated with each line of the selected cache set) to determine a hit or miss and make the final line selection.

Instruction cache. The 8-Kbyte, two-way, set-associative instruction cache is organized into 128 sets, with two lines for each set, and 32 bytes (eight instructions) in each line. The two-way set associativity gives the cache substantially better hit rates than direct-mapped caches at the 8-Kbyte size, and, due to the implementation techniques used, does not adversely affect clock cycle time. The cache has a one-clock-cycle access time and can provide a pair of instructions (64 bits) to the instruction dispatch unit on each cycle, regardless of whether the access is aligned to an odd or an even word address. The only time the cache fails to deliver two instructions is when the instruction pair straddles a cache-line boundary. In practice this turns out to have a very minor performance impact.

We designed the cache to minimize latency on a miss. Figure 20 shows the hardware involved in a cache miss. On a miss, the processor initiates an eight-word burst transaction on the bus to fill the cache line. The burst can transfer two instructions from the bus into the cache on each clock cycle.

The burst begins with the missed instruction pair, continues transferring to the end of the cache line, and then wraps around to fill the beginning of the cache line (if necessary). As soon as the cache receives the missed instruction from the bus, it forwards the instruction directly to the instruction unit so that execution can resume immediately. As the cache receives subsequent instructions from the bus, it also streams them directly into the instruction unit so that execution doesn't stall while the cache line is being brought in from memory.

As shown in Figure 21, the cache is arranged in eight 1-Kbyte blocks; each block is 64-bits wide by 128 rows. Each row contains one 32-bit word from each of the two cache lines. Four of the cache blocks hold the even words of a pair; the other four hold the odd words. When access is made to an evenly aligned instruction pair, the least significant word returns on the even bus and the most significant word returns on the odd bus. When access is made to an oddly aligned pair, the least significant word returns on the odd bus and the most significant on the even bus. The instruction sequencer swaps

the two words before using them.

It is the responsibility of software to maintain instruction cache coherency. Thus, when the virtual memory system swaps in a new page, the instruction cache entries may no longer be valid because a particular logical address may now map to a different physical address. The 88110 provides a fast (approximately five clock cycles) cache-invalidate and cache-line-invalidate feature that enables supervisor routines to eliminate stale data from the cache. Instruction cache coherence with other caches in a multiprocessor system is not normally a problem because processors do not frequently write into instruction space. In fact, the 88110 hardware does not directly support self-modifying code—that is, a program that writes into the currently executing instruction stream. However, the operating system does need to implement loaders, computed programs, copying garbage collectors, and other such programs.

Data cache. The data cache's organization resembles that of the instruction cache's. It is 8 Kbytes in size, two-way set associative, and has eight words in each line. It has a single-clock-cycle access time and can provide 64 bits each clock cycle to the load/store unit. The normal cache-write policy is "store-in" (write-back with write-allocate). And, as with the instruction cache,

burst line fills begin on the missed word and data is forwarded and streamed off the bus, through the load/store unit, and directly into the register files to minimize miss latency.

We selected store-in policy because bus traffic is less for store-in caches than for store-through caches.²³ In store-through caches, the number of main memory references is never less than the store frequency regardless of cache size.²⁴ Considering that 20-30 percent of memory references are typically stores, this can be a problem. Store-in policy helps in multiprocessor systems, where bus utilization must be kept low for good system performance.

On a load or store that hits the cache, the memory reference accesses the cache directly. On a miss, cache control logic selects a line in the cache for replacement, using a pseudorandom selection algorithm that gives priority to invalid lines. If the line selected for replacement has not been modified since being brought into the cache, the hardware simply brings in a new cache line from memory to overwrite it. If the selected line has been modified, it is first copied back to memory before the new line comes in to replace it. A store that hits the cache on a clean line, writes directly into the cache and also broadcasts a message to other processors on the bus to invalidate any copies of this cache line they may have in their local caches.

The hardware automatically maintains data cache coherence. The data cache employs the four-state MESI cache coherency protocol illustrated in Figure 22 on the next page. A write-invalidate procedure guarantees that only one processor on the bus has a modified copy of any given cache line at the same time.

The coherency protocol is enforced by bus snooping, whereby each processor watches (snoops) all bus transactions to track the proper state for each cache line.²⁵ For example, if a bus transaction occurs for a cache line that a processor happens to have in the modified state, it forces the originator of the transaction off the bus, copies the modified line back to memory, changes the state of its line to shared-unmodified, and then allows the original bus transaction to be retried. The cache maintains a separate set of address and state tags for snooping, so that bus snooping does not interfere with the processor's access to its local cache.

Although hardware fully maintains data cache coherence from a multiprocessor point of view, the operating system must still flush stale data out of the cache when the virtual memory map is altered. The data cache can be invalidated quickly on a

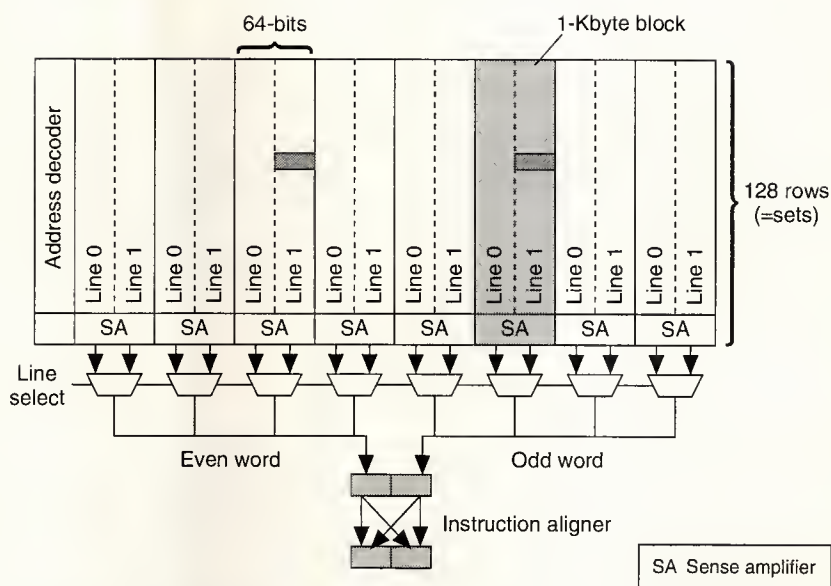


Figure 21. Cache organization.

line or entire-cache basis. The cache can be cleaned (copy-back of dirty lines) or flushed (copy-back of dirty lines with invalidation) on a line, page, or entire-cache basis. The operating system activates invalidation and flushing operations by writing commands to cache control registers accessible only from supervisor mode. Invalidation operations are very fast, requiring approximately five clock cycles for either line or full-cache invalidations. Cleaning or flushing on a page or entire-cache basis requires one clock cycle for each cache set (each cache contains 128 sets) plus the memory transfer time needed to copy back any dirty lines.

External bus interface

The 88110 processor has a high instruction throughput and therefore generates a high rate of memory accesses. The on-chip caches provide relatively high hit rates and eliminate most off-chip memory accesses, but even so a substantial amount of external memory traffic can occur. To keep bus usage down to a point that a tightly coupled, dual-processor system is viable, we used a store-in (write-back) data cache policy to reduce bus traffic and also developed an efficient multiprocessor bus.

The 320-Mbyte/s, synchronous, demultiplexed, pipelined bus supports a retry cache coherency snooping protocol and offers burst-mode and split-transaction transfers. A 64-bit data path minimizes data transfer time on the bus; burst-mode cache-line fills reduce transaction overhead; a split-transaction protocol allows other masters to use the bus while another waits for memory; and address pipelining allows memory

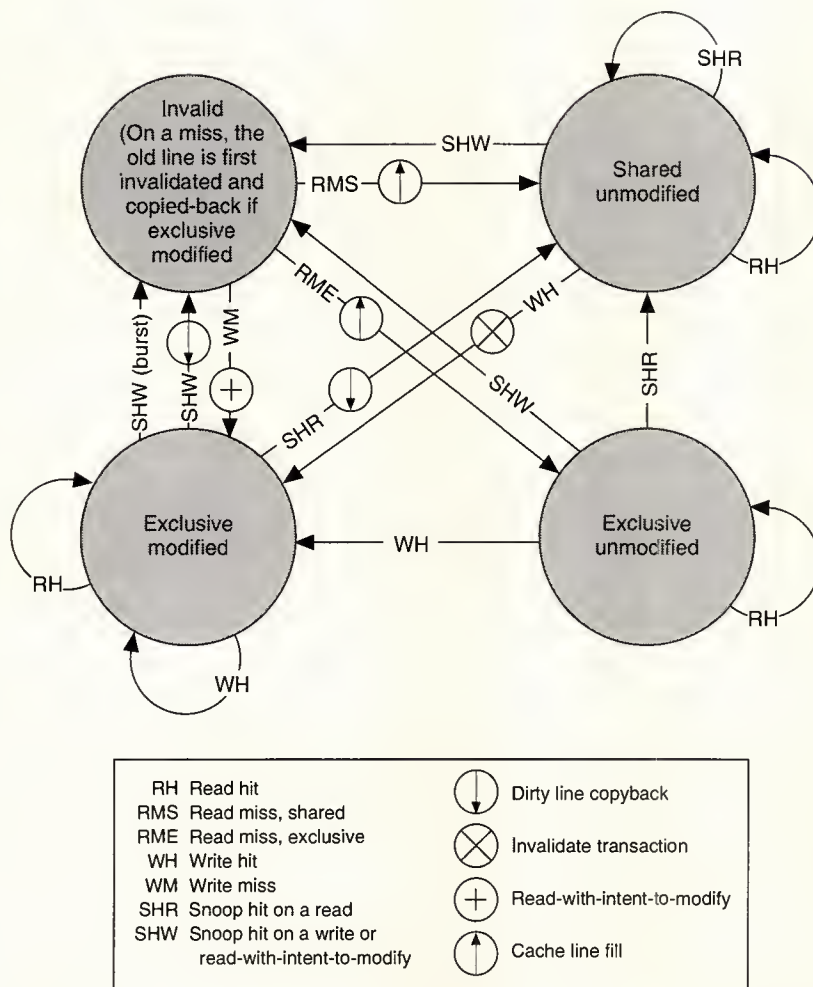


Figure 22. MESI cache coherency protocol.

access time to overlap data transfer time. These features, along with the snoop data cache, make simple, low-cost multiprocessor systems practical.

Bus arbitration, handshaking, and data transfers are all synchronous with the system clock and are referenced to a single clock edge. An internal, analog, phase-locked loop circuit minimizes skew between the internal clock cycle and external signals referenced to the system clock. This circuit greatly simplifies the problem of electrically interfacing to the chip at high speed.

A centralized controller uses a simple bus-request/grant protocol to arbitrate bus ownership. The arbiter may "park" a processor on the bus to eliminate arbitration latency in the frequent cases that bus ownership does not change between successive transfers.

The 32-bit address bus is separate from the 64-bit data bus

to support address pipelining. Address pipelining allows the address phase of a bus transaction to run concurrently with a previous data transfer phase. In multiprocessor configurations, address pipelining allows memory access times to overlap data transfer times, thereby increasing available bus bandwidth.

In the past, most microprocessor buses used a tenured transaction protocol. A tenured protocol ties up the bus from the time a transaction starts until the entire memory cycle completes and data returns to the processor. In a DRAM system with relatively long access time, this protocol wastes considerable bus bandwidth. The 88110, on the other hand, uses a split-transaction bus protocol, which allows a bus transaction to be split into distinct address and data phases that are controlled independently.

For example, a processor can send an address request to the memory system and then permit other processors to use the bus while it waits for a response. This protocol uses the bus more efficiently, consuming bandwidth only during the time addresses or data actually transfer. Address pipelining and split transactions permit the 88110 to more closely approach the theoretical bus bandwidth limit than microprocessors that use tenured bus protocols.

All burst-mode transactions transfer an entire cache line. A burst transfer uses four data beats; each beat transfers 8 bytes of data. The system controls the length of time

of each data beat, which can be as short as one clock cycle.

The diagram in Figure 23 shows a possible sequence of bus transactions in a multiprocessor system (for clarity some control signals have been omitted). On the first clock cycle shown, a processor (CPU A) is parked on the bus (BG A preasserted) and requests a data cache line fill by asserting Transfer Start (TS) and driving the address bus. Two clock cycles later, the memory acknowledges receipt of the address (AACK), and CPU A terminates its address phase. Meanwhile, a second processor (CPU B) has asserted Bus Request (BG B) for an instruction cache line fill and has been scheduled next onto the bus by the arbiter's assertion of Bus Grant (DBG B) to it. As soon as CPU A relinquishes the address bus, CPU B starts its cycle. In this example, CPU B's request gets serviced immediately by memory, and the arbiter allows it to read data by granting it access to the data bus (DBG B asserted). The data

transfer here is shown occurring at full bus speed; however, the memory system could use Transfer Acknowledge (TA) to pace the transfer by inserting wait states on each data beat. As soon as the transfer to CPU B finishes, the arbiter grants data bus to CPU A (DBG A) for its data transfer.

The bus-snooping mechanism, illustrated in Figure 24, enforces cache coherency among all processors on the bus. When a processor (in this example, CPU A) that is currently the bus master puts a global (GBL) address out on the bus, all other processors on the bus snoop the address. If one of these processors (CPU B in this case) has a modified copy of the data being requested in its local cache, it signals that fact to the requesting processor (CPU A) via a snoop status signal (SSTAT B). Upon seeing the snoop hit signal, CPU A aborts its bus transaction and relinquishes control. The snooping processor (CPU B) then takes control of the bus and copies its modified line back out to memory. When this transaction is complete, CPU A retries the original transaction, which now completes normally since all caches are consistent with memory. The control signals are flexible enough to support more sophisticated protocols such as intervention with direct cache-to-cache transfer (snarfing).⁶

System features

The 88110 includes several features designed to improve system debugging, reliability, and testability.

One of the few drawbacks of on-chip caches is that they filter external memory references, which limits visibility and reduces the utility of in-circuit emulators for software debugging. One software debug issue, for example, is detecting the source of corruption of a particular program variable or data structure. The 88110 addresses this problem by providing two data breakpoint registers that allow a program to trap to a software debugger on an access to, or modification of, a specified logical address or range of addresses (byte, half, word, double, quad, ..., page). The 88110 also has a facility that allows a debugger to single-step a program one instruction at a time.

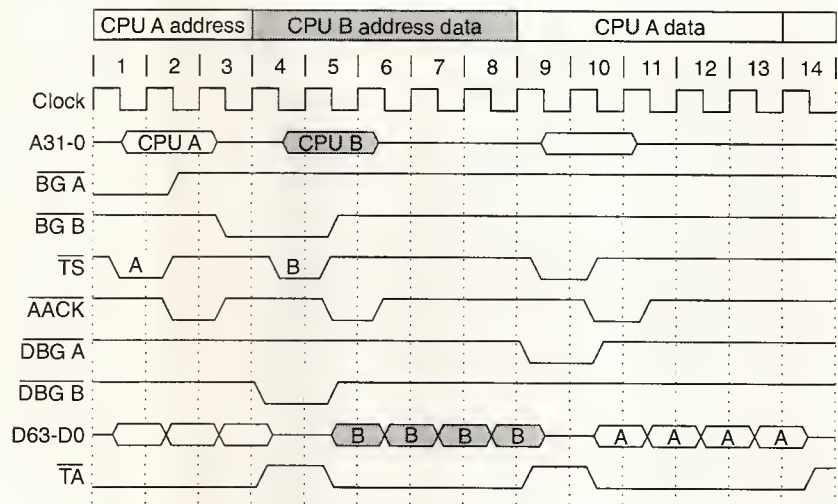


Figure 23. Split bus transaction.

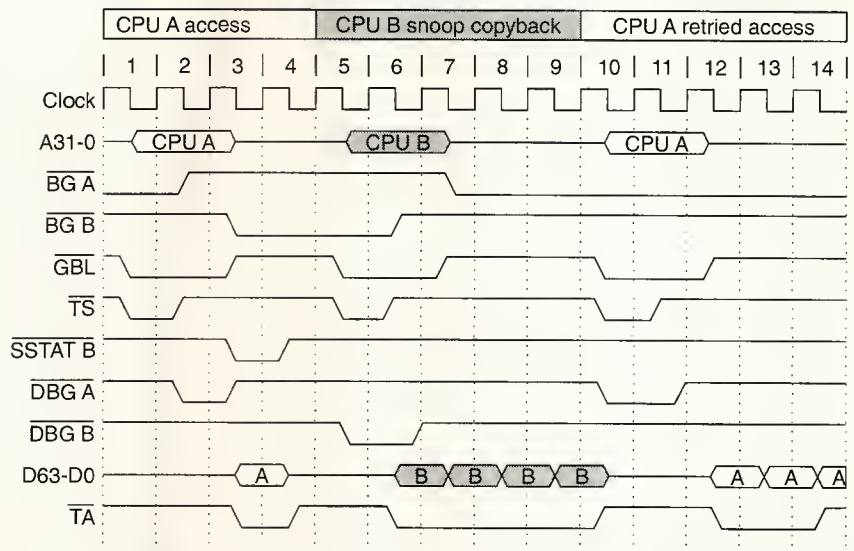


Figure 24. Retry bus-snooping protocol.

Two features improve the 88110's applicability in high-reliability systems: data bus parity and deterministic lockstep operation. On all external data bus write operations, the processor generates an odd parity bit for each byte transferred on the bus. On data bus read operations, the processor checks the parity bits and generates an interrupt if any byte transfers in error. For redundant systems, lockstep operation makes it possible to shadow one 88110 with another; once synchronized, two 88110s will stay in lockstep so long as they are

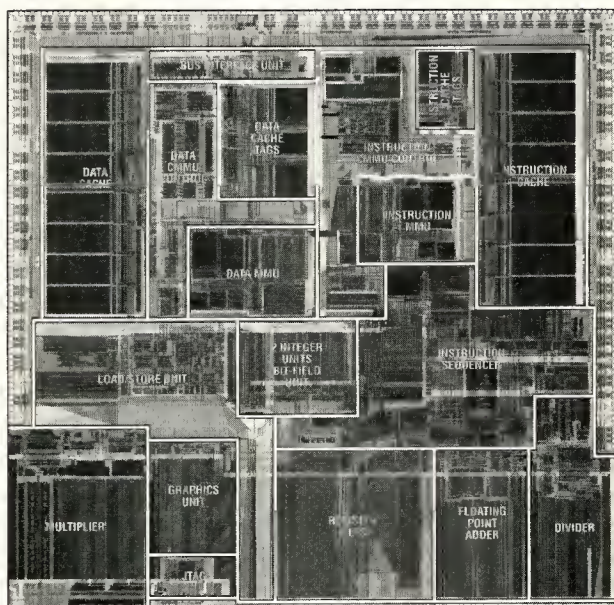


Figure 25. Photograph of 88110 die.

presented with the same inputs at the same time.

We improved the in-system testability of the 88110 by providing JTAG/IEEE 1149.1 boundary scan logic on all relevant I/O pins.

Silicon

We designed the 88110 in a triple-level metal, double-level polysilicon CMOS process. We used 1- μm design rules with transistor channel lengths reduced to an effective length of less than 0.8 μm . The die is easily shrinkable to 0.8- μm (0.65- μm effective channel length) technology without design modifications. The complete design required less than 1.3 million transistors and fits on a die 15 μm on a side (Figure 25). The cache SRAM cells are a four-transistor, NMOS, polyload, bit-cell design, which uses a P-well process for high immunity to soft errors.

Initially, we plan to provide the 88110 in a through-hole, ceramic pin grid array package. The 299-pin, 20×20 , cavity-down package measures approximately 2 inches on a side, with 100-mil pin pitch.

Performance

Official benchmark data from real systems and production compilers is not available at the time of this writing. However, a good-quality prototype optimizing compiler (the Motorola 88110 Alpha compiler) is available, as well as a clock-for-clock instruction simulator that accurately models all processor pipeline, primary cache, TLB, and memory sys-

Table 1. Simulated SPEC ratios at 50 MHz.

Benchmark	Ratio
Gcc	46.5
Espresso	48.1
LI	57.0
Eqntott	52.9
Spice2g6	34.7
Doduc	41.4
Nasa7	67.9
Matrix300	357.8
Fpppp	64.4
Tomcatv	72.2
Geometric means	
Integer	51.0
Float	73.9
Combined	63.7

tem effects. Results indicate a Dhrystone 2.1 performance that translates to well over 100 VAX MIPS.

We also used the instruction simulator to run the SPEC benchmark suite at 50-MHz with a 180-ns (9/1/1) DRAM memory system. The results appear in Table 1. These benchmarks were compiled with the Motorola 88110 Alpha compiler, except LI, which was compiled with the Diab 88110 Compiler Version 2.37. The Nasa 7 and Matrix 300 benchmarks were preprocessed by the Kuck and Associates preprocessor. In a recent publication, Mike Phillip reports more completely on the 88110 compilers and performance.²⁶

The instruction simulator does not yet accurately model all effects of external secondary-cache misses, so we haven't reported results with a second-level cache here. However, simulations with an infinite secondary cache show a combined Specmark above 80.

A significant characteristic of the 88110 is that it makes parallel instruction execution fairly easy to achieve in practice. Relatively simple compilers can produce effective code schedules for the 88110; in fact, the processor realizes substantial parallelism even on code originally generated for the 88100 single-issue CPU. The efficiency of superscalar issue ranges from 20 percent to over 50 percent, depending on the benchmark and memory configuration. Currently, over the SPEC benchmark suite, we find that two instructions issue on roughly half (ranging from about 35-70 percent) of the clock cycles on which an instruction executes at all. Of course, we expect these results to continually improve with advances in our compilers.

The increasing use of high-level languages makes good


```

Clock 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
|-|-|-| fmul      |-|-| st      |-|-|-| fadd
|-|-|-| ld        |-|-|-| fmul    |-|-|-| fmul
|-|-|-| fmul      |-|-|-| fadd    |-|-|-| fadd
|-|-|-| ld        |-|-|-| fadd    |-|-|-| fmul
|-|-|-| fadd      |-|-|-|-| ld    |-|-| st
|-|-|-| fmul      |-|-|-|-| fadd  |-|-|-| fmul
|-|-|-|-|-| ld   |-| add        |-|-|-| fadd
|-|-|-| fmul      |-| add        |-|-|-| fmul
|-|-|-| fadd      |-|-| st       |-|-|-| fadd
|-|-|-| fmul      |-|-|-| fmul    |-|-|-| fmul
|-|-|-| fmul      |-| sub       |-|-|-| fadd
|-| add           |-|-|-| fmul    |-|-|-|-| fmul
|-|-|-| fadd      |-|-| ld       |-|-| st
|-|-|-| fmul      |-|-|-| fmul    |-|-|-|-| fmul
|-|-|-| fadd      |-|-| ld       |-|-|-| fadd
|-|-|-| fmul      |-|-|-| fmul    |-| add
|-|-|-| fadd      |-|-| ld       |-|-|-| fadd
|-|-|-| fmul      |-|-|-| fadd    |-|-|-|-| ld
|-|-| st          |-|-|-| fmul    |-|-|-| fadd
|-|-|-| fmul      |-|-|-| fadd    |-|-|-| fmul
|-|-|-| fadd      |-|-|-| fmul    |-| add
|-|-|-| fmul      |-|-|-| fadd    |-|-| st
|-|-|-| fadd      |-|-|-| fmul    |-|-|-| fadd
|-|-|-| fmul      |-|-| st       |-|-|-|-| st
|-|-|-| fadd      |-|-|-| fmul    |-|-|-|-| st
|-|-|-| fmul      |-|-|-| fadd    |-| bcd.n
|-|-|-| fmul      |-|-|-| fmul    |-| add

```

performance on general compiled code essential. But many programs spend a great deal of time in a few critical routines. System response time can dramatically improve by tuning a few of these hot spots. DSP algorithms for voice processing, graphics library routines, video processing, and interactive user interface routines are prime candidates for this type of tuning. As an example, the double-precision floating-point matrix multiply routine often used in graphics viewpoint transformations is illustrated in Figure 26. On this code the 88110 can issue two instructions on nearly every clock cycle and can sustain 97.5 MIPS and 68 double-precision Mflops (at 50 MHz), even if the point vectors being transformed are not in the cache and the processor is operating into DRAM.

Although the hit rate of the 88110's internal caches is quite high, long DRAM latency and high bus utilization can still limit performance. For ultimate performance, or for system designs calling for more than two tightly coupled processors, we must further reduce memory access time and bus use.

grated second-level cache, consisting of the 88410 cache controller and an array of 62110 cache SRAMs. We implemented the secondary-cache function as a separate chip, rather than putting the logic on the 88110 itself, so that low-end systems don't have to pay for transistors they don't use.

The diagram illustrates the system architecture. At the top is the MC88110, which is connected to the MC88410 below it via an Address bus and a Data bus. The MC88410 is connected to a 256 Kbytes memory array (consisting of eight 62110 chips) via an Address bus. The memory array is connected to the System address bus and the System data bus.

April 1992 61

control functions and tags for 256 Kbytes to 1 Mbyte of cache. Cascaded 88410s can support larger cache sizes. The cache tags included on the 88410 allow all hit, miss, and data-steering decisions to be made quickly without accessing off-chip SRAMs. This approach also reduces pin count and the number of SRAM packages required, thereby minimizing the system cost.

The SRAM cache array sits directly in the 88110 data bus path and the 88410 controls all data transfers into, out of, and through the array. The 62110 cache SRAM device works especially well in an 88110/410 secondary-cache system. The 62110, based on a standard 32K × 9, 12-ns SRAM, has a dual-bus architecture that allows data to be fed directly from the system bus onto the 88110 data bus and simultaneously captured in the internal SRAM array. We plan to offer the 62110 commercially as a cache part for other systems as well.

The secondary cache implemented by the 88410 is a direct-mapped cache with a store-in (write-back) write policy. Line length is configurable to either 32 or 64 bytes. Cache hits, using the 62110 SRAM, present a 3/1/1/1 memory cycle to the 88110.

A four-state, MESI protocol enforced by bus snooping maintains horizontal coherency between the 88410 cache and other caches on the system bus. The 88410 uses inclusion²⁸ to maintain vertical coherency between the 88110's internal cache and the secondary-cache array.

The bus protocol and electrical interface used by the 88410 are similar to those used by the 88110. As a result, one can design a system that can accept either an 88110 or an 88110/88410/62110 module. The 88410 also has an option to allow the system bus to operate at half the speed of the 88110 bus. This feature relaxes system timing constraints and will eventually allow systems to accommodate higher frequency 88110/410 modules, using standard TTL electrical interfaces.

IN DESIGNING THE 88110, our goal was to produce a high-performance, general-purpose microprocessor at a cost consistent with use in low-cost personal computers and workstations. We accomplished this goal with an advanced superscalar architecture and a high level of circuit integration implemented in a fine-geometry, high-yield, semiconductor fabrication process. ■

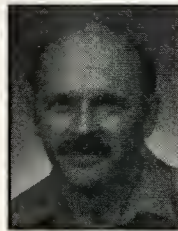
Acknowledgments

The 88110's performance and innovation resulted from the efforts of many people on the design and support teams. Bill Moyer provided technical leadership on the circuit design. Dave Mothersole, Jim Klingshirn, and Bill Moyer skillfully directed the design team. Similar acknowledgment goes to the software development team led by Anne-Marie Larkin, the 88410 design team led by Russell Stanphill, and the 62110 design team led by Brian Peterson. Special mention goes to Mike Phillip for his technical expertise in making a world-class compiler available for the 88110. Thanks also to Yoav Talgam and Mitch Alsup for their seminal work on the original 88000 architecture, which greatly simplified the task of building a high-performance superscalar microprocessor.

References

1. T. Agerwala and J. Cocke, "High-Performance Reduced Instruction Set Processors," IBM Tech. Report, N.Y., Mar. 1987.
2. G.M. Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities," *Proc. AFIPS 1967 Spring Joint Comp. Conf.*, Apr. 1967, pp. 483-485.
3. J.A. Fisher, "Very Long Instruction Word Architectures and the ELI-512," *Proc. 10th Annual Int'l Symp. Computer Architecture*, June 1983, pp. 140-150.
4. J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publ., San Mateo, Calif., 1990.
5. N. Jouppi and D. Wall, "Available Instruction-Level Parallelism for Superscalar and Superpipelined Machines," *Proc. Third Conf. Architectural Support for Programming Languages and Operating Systems*, ACM, Apr. 1989, pp. 272-282.
6. *Futurebus+, Logical Layer Specification, Draft 8.02, P896.1/ D8.02*, CS Press, Los Alamitos, Calif., Sept. 1989.
7. *Systems Performance Evaluation Cooperative, SPEC Fact Sheet*, Waterside Associates, Fremont, Calif., 1989.
8. *ANSI/IEEE 754-1985 Standard for Binary Floating-Point Arithmetic*, IEEE, Piscataway, N.J., 1985.
9. G. Radin, "The 801 Minicomputer," *Proc. Symp. Architectural Support for Programming Languages and Operating Systems*,

- ACM, Mar. 1982, pp. 39-47.
10. D.A. Patterson and C.H. Sequin, "A VLSI RISC," *Computer*, Vol. 15, No. 9, Sept. 1982, pp. 9-21.
 11. R.M. Tomasulo, "An Efficient Algorithm for Exploiting Multiple Arithmetic Units," *IBM J. Research & Development*, Vol. 11, No. 1, Jan. 1967, pp. 25-33.
 12. J. Smith, "Dynamic Instruction Scheduling and the Astronautics ZS-1," *Computer*, Vol. 22, No. 7, July 1989, pp. 21-35.
 13. S. Weiss, and J. Smith, "Instruction Issue Logic for Pipelined Supercomputers," *Proc. 11th Annual Int'l Symp. Computer Architecture*, IEEE/ACM, 1984, pp. 110-118.
 14. J.E. Thornton, *Design of a Computer—The Control Data 6600*, Scott, Foresman, and Co., Glenview, Ill., 1970.
 15. D. Lilja, "Reducing the Branch Penalty in Pipelined Processors," *Computer*, Vol. 21, No. 7, July 1988, pp. 47-55.
 16. D.A. Patterson and C.H. Sequin, "RISC-1: A Reduced Instruction Set VLSI Computer," *Proc. Eighth Annual Int'l Symp. Computer Architecture*, May 1981, pp. 443-457.
 17. J. Smith and A. Pleszkun, "Implementing Precise Interrupts in Pipelined Processors," *IEEE Trans. Computers*, Vol C-37, No. 5, May 1988, pp. 562-573.
 18. D. Kroft, "Lockup-free Instruction Fetch/Prefetch Cache Organization," *Proc. Eighth Int'l Symp. Computer Architecture*, May 1981, pp. 81-87.
 19. C. Chi and H. Dietz, "Improving Cache Performance by Selective Cache Bypass," *Proc. 22nd Annual Hawaii Int'l Conf. on Systems Sciences*, Vol. 1, CS Press, 1989, pp. 277-285.
 20. M. Dubois, C. Scheurich, and F. Briggs, "Synchronization, Coherence, and Event Ordering in Multiprocessors," *Computer*, Vol. 21, No. 2, Feb. 1988, pp. 9-21.
 21. T. Kilburn et al., "One-Level Storage System," *IRE Trans. Electronic Computers*, Vol. EC-11, IEEE Service Center, Apr. 1962, pp. 223-235.
 22. D.L. Black et al., "Translation Lookaside Buffer Consistency: A Software Approach," Carnegie Mellon Univ., Tech. Report CMU-CS-88-201, Pittsburgh, Dec. 1988.
 23. R.L. Norton and J.L. Abraham, "Using Write-Back Cache to Improve Performance of Multiuser Multiprocessors," *Proc. Int'l Conf. Parallel Processing*, 1982, pp. 326-331.
 24. J. Bell, D. Casasent, and C.G. Bell, "An Investigation of Alternative Cache Organization," *IEEE Trans. Computers*, Vol. C-23, No. 4, Apr. 1974, pp. 346-351.
 25. R.H. Katz et al., "Implementing a Cache Consistency Protocol," *Proc. 12th Annual IEEE Symp. Computer Architecture*, 1985, pp. 276-283.
 26. M. Phillip, "Performance Issues for the 88110 RISC Microprocessor," *Proc. Compton*, Feb., 1992.
 27. J. Goodman, "Using Cache Memory to Reduce Processor-Memory Traffic," *Proc. 10th Annual Int'l Symp. Computer Architecture*, June 1983, pp. 124-131.
 28. W.H. Wang, J.L. Baer, and H. Levy, "Organization and Performance of a Two-Level Virtual-Real Cache Hierarchy," *Proc. 16th Annual Symp. Computer Architecture*, CS Press, 1989, pp. 140-148.



Keith Diefendorff, a member of the technical staff in Motorola's RISC Division in Austin, Texas, held responsibility for the definition of the 88110. He currently works with Apple and IBM architects to define the Power PC RISC architecture and investigates advanced microarchitecture concepts for future processors. Previously, he was a senior member of the technical staff at Texas Instruments where he designed ICs and was the primary system architect of the Explorer and micro Explorer symbolic processing LISP workstations.

Diefendorff holds an MSEE from the University of Akron. He is a member of the IEEE and the Computer Society.



Michael Allen, a senior systems designer in the same RISC Division, helped define the 88110 bus interface. He currently works with Apple and IBM to define the bus interfaces for Power PC-based microprocessors. Before joining Motorola, he designed board-level systems and worked as an architect for the DP8344 RISC integrated microprocessor at National Semiconductor. Allen holds a BSEE from the University of Texas at Arlington.

Direct questions concerning this article to Keith Diefendorff, Motorola Inc., MDOE215, 6501 William Cannon Drive West, Austin, TX 78735; keithd@oakhill.sps.mot.com.

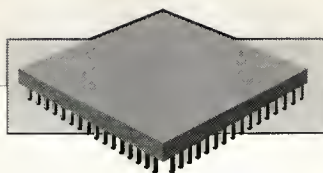
Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

High 159

Medium 160

Low 161



The Proposed SSBLT Standard Doubles the VME64 Transfer Rate

A revision to the IEEE 1014 VMEbus standard will offer a source synchronous block transfer protocol that doubles the transfer rate without changing the backplane or electrical interface. The faster rate in turn doubles the performance/cost ratio of the bus.

Jack Regula

Force Computers

In 1991, the IEEE P1014R (Revision D) working group drafted a new transfer mechanism for the 64-bit VMEbus:¹ the source synchronous block transfer (SSBLT). The working group gave preliminary approval to the SSBLT protocol as described here; it is thus on its way to becoming an IEEE standard.

With SSBLT, the source of the data supplies the clock used to sample the data at the destination. Consequently, the working group applied the term *source synchronous block transfer* to the protocol. SSBLT achieves higher performance by eliminating the protocol delays built into the original VMEbus specification. It is optimized by its source synchronous nature, which minimizes the skew between the data and the clock.

SSBLT doubles the rate at which data transfers between masters and slaves. Operating over the 64-bit VMEbus (as defined in the latest proposed draft, Rev. D), data transfers at 20M transfers per second times 8 bytes per transfer, for a burst transfer rate of 160 Mbytes/s.

Significantly, this performance improvement results purely from protocol improvements. SSBLT allows transfers to make use of standard VMEbus backplanes and driver technology and permits systems employing SSBLT to be backward com-

patible with present IEEE 1014 VMEbus modules.

Progress

VMEbus performance has increased in several steps since it was introduced 10 years ago. From the original maximum transfer rate of 10 to 20 Mbytes/s without block transfers, VMEbus throughput increased to a peak of 30-40 Mbytes/s when using 32-bit block transfers. Block transfers raise performance levels because, after an initial access latency, many slaves can supply data at a higher rate. VMEbus handshaking and protocol delays limit this rate to something less than 10M transfers per second.

Multiplexed block transfers (MBLTs) were proposed about three years ago and began reaching production status during 1991. MBLTs double block transfer performance by doubling the data path width. But, since they use the same protocol and timing rules as block transfers, MBLTs are also limited to less than 10M transfers/s.

MBLTs employ address/data multiplexing to double the data path width and, optionally, the address width. During the first cycle of an MBLT, which is conveniently called the address phase, no data transfers over the bus. In addition, another 32 bits of address are multiplexed onto the



April 1992 issue (card void after October 1992)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest

(Add comments on the back)

Readers, Indicate your interest in articles and departments by circling the appropriate number (shown on the last page of articles and departments):

150 151 152 180 181 182
153 154 155 183 184 185
156 157 158 186 187 188

159 160 161 189 190 191
162 163 164 192 193 194
165 166 167 195 196 197

168 169 170 198 199 200
171 172 173 201 202 203
174 175 176 204 205 206

177 178 179 207 208 209

Product Information

(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	



April 1992 issue (card void after October 1992)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest

(Add comments on the back)

Readers, Indicate your interest in articles and departments by circling the appropriate number (shown on the last page of articles and departments):

150 151 152 180 181 182
153 154 155 183 184 185
156 157 158 186 187 188

159 160 161 189 190 191
162 163 164 192 193 194
165 166 167 195 196 197

168 169 170 198 199 200
171 172 173 201 202 203
174 175 176 204 205 206

177 178 179 207 208 209

Product Information

(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	

SUBSCRIBE TO IEEE MICRO

All the facts about today's chips and systems

☐ **YES**, sign me up!

If you are a member of the Computer Society or any other IEEE society, pay the member rate of only \$23 for a year's subscription (six issues).

Society: _____

IEEE membership no: _____

Society members: Subscriptions are annualized. For orders submitted March through August, pay half the full-year rate (\$11.50) for three bimonthly issues.

Full Signature _____ Date _____

Name _____

Street _____

City _____

State/Country _____ ZIP/Postal Code _____

☐ **YES**, sign me up!

If you are a member of ACM, ACS, BCS, IEE (UK), IEEE but not a member of an IEEE society, IECEJ, IPSJ, NSPE, SCS, or other professional society, pay the sister-society rate of only \$39 for a year's subscription (six issues).

Organization: _____ Membership no: _____

For airmail option, see page 2.

☐ Payment enclosed Residents of CA, DC, Canada, and Belgium add applicable sales tax.

☐ Charge to ☐ Visa ☐ MasterCard ☐ American Express

Charge-card number _____

Expiration date _____

Month _____ Year _____

Prices valid through 12/31/92
04/92 MICRO

Charge orders also taken by phone:
(714) 821-8380 8 a.m. to 5 p.m. Pacific time
Circulation Dept.
10662 Los Vaqueros Circ., PO Box 3014
Los Alamitos, CA 90720-1264

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

Reviewers Needed. If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.

For reader service inquiries, see other side.



PLACE
POSTAGE
HERE

PO Box is for reader service cards only.

IEEE Micro

PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA

Editorial comments

I liked: _____

I disliked: _____

I would like to see: _____

Reviewers Needed. If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.

For reader service inquiries, see other side.



PLACE
POSTAGE
HERE

PO Box is for reader service cards only.

IEEE Micro

PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

IEEE COMPUTER SOCIETY

PO BOX 3014
LOS ALAMITOS CA 90720-9804
USA



data lines whenever the A64 mode (64 bits of address) is in use. After the first DTACK signal assertion (DTACK*), both the address and data lines can be used for data. From this point on, the timing for MBLTs is the same as for 32-bit block transfers. Therefore, performance doubles with MBLTs.

The 64-bit address capability added to VMEbus with MBLTs also significantly extended its useful life. And, to address the increased use of bus bridges in future systems, Revision D for SSBLTs adds a cycle retry function intended to allow deadlocks to be broken. All these enhancements are compatible with or variations of the original, asynchronous, four-edge, strobe-acknowledge VMEbus handshake.

***The key to the SSBLT's ability to
double transfer rates is its
elimination of several protocol
delays included in the original
VMEbus standard.***

The SSBLT mechanism goes beyond that of the MBLT by eliminating the strobe acknowledge handshake that limits the performance of the asynchronous protocol. Like the MBLT, SSBLT multiplexes address and data lines to form a 64-bit data path. The address phase is identical to that of the MBLT and can include either 32 or 64 bits of address. But in the data transfer portion, data is clocked from source to destination without cycle-by-cycle handshaking at a rate of up to 20 MHz or 160 Mbytes/s.

SSBLTs contain unique address modifier codes: 07 indicates an A32 SSBLT, and 06 indicates an A64 SSBLT. Boards not capable of performing SSBLTs don't respond to these address modifiers nor assert bus error signal BERR*. Thus the master can repeat the access with another transfer method such as a standard block transfer or an MBLT. This level of interoperability is assured by requiring an SSBLT board to support all earlier transfer methods.

Transfer and throughput rates

Because the cycle-by-cycle handshake has been eliminated, boards can relatively easily transfer data at the peak transfer rate. Contrast this with VMEbus asynchronous handshaking, which is hard pressed to approach 10 MHz and is slowed down by backplane and driver propagation delays. The initial access latency amortized over the entire burst primarily determines data throughput for an SSBLT.

I've estimated that, with back-to-back transfers of 64 bytes, the sustained transfer rate using SSBLT is 128 Mbytes/s for writes and 100 Mbytes/s for reads. Increasing the block size to 2 Kbytes boosts the estimated sustained rate to 159 and 157 Mbytes/s for writes and reads.

The sustained-rate calculations assume 100 ns for the address phase on a write transfer and 240 ns for reads, including initial access latency (typical of high-performance VMEbus interfaces). Because 8 bytes transfer every cycle, each 64-byte block requires eight transfers. At the SSBLT maximum rate, transfers execute every 50 ns. Therefore, the calculations for back-to-back 64-byte blocks are

Write transfers

$$100 \text{ ns} + 8 \text{ transfers} \times 50 \text{ ns} = 500 \text{ ns/block}$$
$$64 \text{ bytes}/500 \text{ ns} = 128 \text{ Mbytes/s}$$

Read transfers

$$240 \text{ ns} + 8 \text{ transfers} \times 50 \text{ ns} = 640 \text{ ns/block}$$
$$64 \text{ bytes}/640 \text{ ns} = 100 \text{ Mbytes/s}$$

When the block size increases to 2 Kbytes, requiring 256 transfers to complete, the overhead of the address phase is amortized over a larger data transfer period. Thus, back-to-back transfers of the larger blocks yield

Write transfers

$$100 \text{ ns} + 256 \text{ transfers} \times 50 \text{ ns} = 12,900 \text{ ns/block}$$
$$2 \text{ Kbytes}/12,900 \text{ ns} = 159 \text{ Mbytes/s}$$

Read transfers

$$240 \text{ ns} + 256 \text{ transfers} \times 50 \text{ ns} = 13,040 \text{ ns/block}$$
$$2 \text{ Kbytes}/13,040 \text{ ns} = 157 \text{ Mbytes/s}$$

The write latency in these calculations assumes that the packet can be received without delay at the beginning of the transfer's data cycle. The calculations also assume that, for reads of small blocks, a FIFO queue buffers the transfers and is partially loaded before the start of a transfer. We estimate this step to require 240 ns. For large block transfers, the calculations assume the circuitry of the boards involved is fast enough to handle the transfer in either direction without throttling.

Eliminating protocol delays

The key to the SSBLT's ability to double transfer rates is its elimination of several protocol delays included in the original VMEbus standard. These delays simplified implementations by allowing architecturally simple interfaces to be implemented with logic that is both agonizingly slow and extremely modest in complexity by today's standards. Today, architectural elegance is affordable, as is subnanosecond logic.

Using the high-speed, high-density ASIC technologies available now, single-chip interfaces—including FIFO buffers and

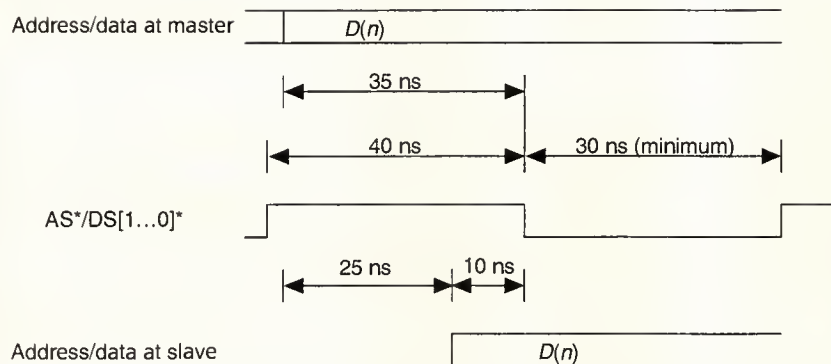


Figure 1. VMEbus protocol delays.

DMA controllers employing a 20-MHz SSBLT protocol—are well within the state of the art. Bus interface ASICs need only a few hundred additional gates to add SSBLT capability to a design that already includes MBLTs.

Protocol design for TTL backplanes is complicated by the considerations of incident wave switching. Incident wave switching² in a transmission line environment refers to a driver's ability to drive its output voltage across the switching threshold of receivers placed along the line as the voltage wavefront first propagates down the transmission line. A driver's incident wave output voltage is reduced by voltage division of its output impedance with the transmission line's impedance. When the driver isn't strong enough for incident wave switching, the switching threshold isn't crossed until a reinforcing reflection arrives from the far end of the transmission line. The resulting waveform then takes on a staircase

appearance with one step per reflection. In VMEbuses, a single step often appears near the threshold region.

TTL backplanes generally do not provide incident wave switching unless they are only lightly loaded. Protocol designers must take into account the possibility that certain signals, such as data strobes, might be received with incident wave switching, while transitions of the data itself might not be seen until a reflection arrives from the far end of the backplane. The original VMEbus standard provided for this situation by requiring the master to provide a 35-ns setup time while guaranteeing the slave

only 10 ns of setup. The difference is two backplane delays totaling 15 ns plus an additional allowance of 10 ns for skew in the bus drivers and receivers. The two backplane delays allow time for the reinforcing reflection to arrive from the far end(s) of the backplane. The SSBLT protocol's data capture delay parameter permits the same effects. At 37.5 ns it is actually slightly more conservative than the original VMEbus protocol! Figure 1 illustrates the VMEbus protocol delays.

VMEbus has two additional protocol delays. The slave may not assert DTACK* until at least 30 ns has elapsed since assertion of DS[1..0]*. Although not shown in Figure 1, the master cannot capture read data from the bus until 25 ns after the assertion of DTACK* because of possible skew between data and DTACK*. These protocol delays mean that even with infinite speed logic and zero-delay backplanes, a compliant VMEbus data transfer cycle takes a minimum of 70 ns for writes

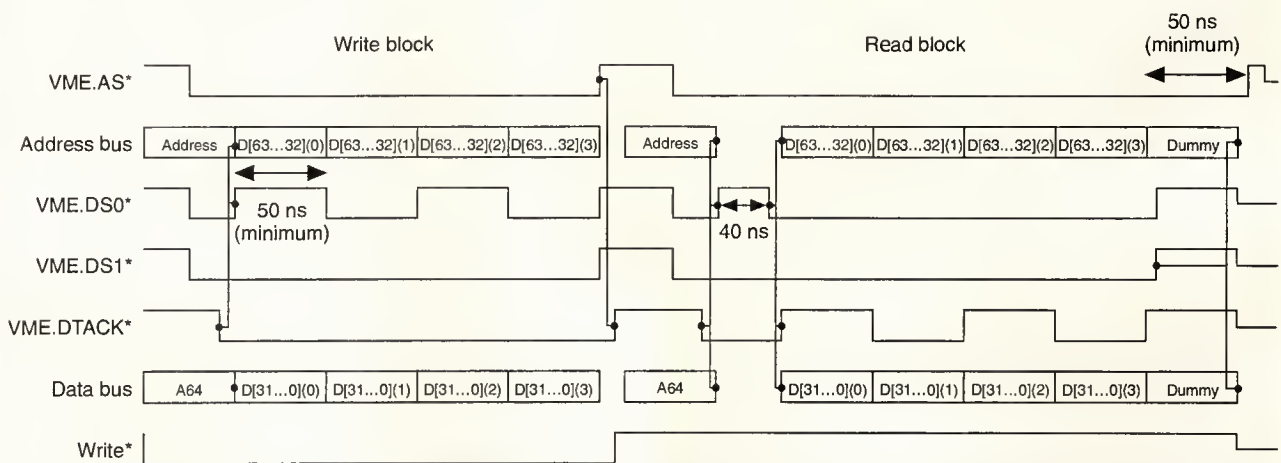


Figure 2. VME64 source-synchronous block transfer. If AS* rises before data capture time, data does not transfer and the cycle ends. The slave can sample AS* at what would have been the data capture time and verify the burst end. DS1* stays asserted throughout the burst to keep the bus timer enabled.

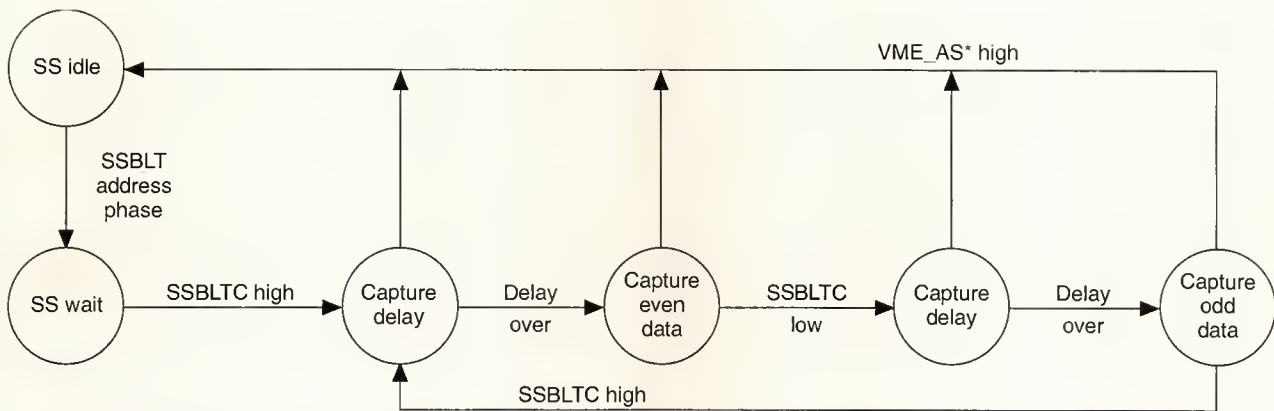


Figure 3. Asynchronous state machine for SSBLT master reads and slave writes. SSBLTC indicates master_read and DTACK* or slave_write and DS0*.

and 95 ns for reads. When practical logic, driver, receiver, and backplane delays are added to these protocol minimums, VMEbus users find it extremely difficult to achieve burst transfer rates of greater than 64 Mbytes/s, even with the MBLT method. The significance of the SSBLT method is that it makes it relatively easy to achieve burst rates of 160 Mbytes/s and to sustain throughputs of over 100 Mbytes/s.

Figure 2 shows both read and write SSBLTs. These contain an address phase in which the slave asserts DTACK* as soon as it recognizes the address and address modifiers and is ready to transfer data. In the write cycle, the master then uses DS0* to clock the data to the slave at a 20-MHz rate (or slower, if desired). In the read transfer, an additional data strobe pulse provides buffer turnaround time. Then the slave, which is the source on a read cycle, uses DTACK* to clock the data to the master.

Figure 3 shows a small asynchronous state machine that may be used by an SSBLT master to receive data on a read cycle or by an SSBLT slave to receive data on a write cycle. Note that in Figure 2 each edge of DS0* or DTACK* transfers data. The SSBLT protocol specifies that at least 50 ns must occur between each edge. The data destination detects each edge, delays a data capture time, then latches the data from the VMEbus. Data is nominally in phase with the clock at the source (± 5 ns). The data capture delay, which must be between 37.5 and 45 ns, allows for nonincident wave switching, 5 ns of skew at each source, and destination and settling times.

Figure 4 illustrates SSBLT protocol

delays. This protocol simply sets the instantaneous transfer rate to the maximum that can be supported (with a margin for safety) and provides the timing rules for data transfer. In contrast with a VMEbus asynchronous handshake, it does not include cycle-by-cycle handshaking delays. Such delays make performance depend upon the physical length of the backplane and the speed of the backplane drivers and interface logic. SSBLT masters and slaves need only keep their skew and capture delay errors within budget to be able to use the maximum transfer rate.

Rescinding DTACK* driver

A rescinding three-state driver is a circuit that is actively driven high and then tristated (changes voltage to high, low, and off states). When used for DTACK*, a rescinding driver speeds up asynchronous block transfers. In a heavily loaded VMEbus backplane, the time constant of the terminators and the distributed capacitance of the bus increases the propaga-

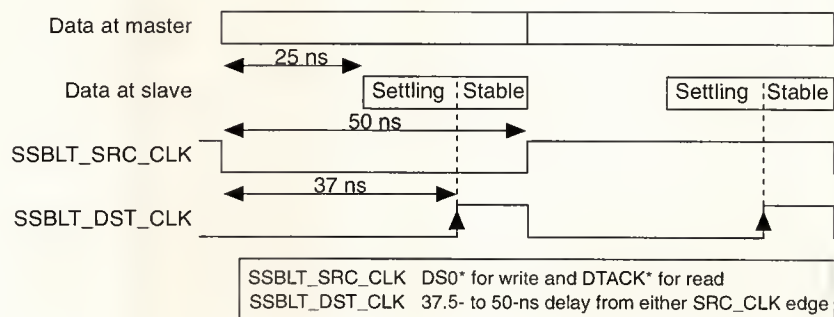


Figure 4. SSBLT protocol delays.

tion delay for the rising edge of DTACK*. The resulting delay is greater than the VMEbus 40-ns minimum strobe, high-time specification. This delays the start of the next cycle.

The SSBLT revision to the VMEbus standard provides the timing rules for use of a rescinding DTACK* driver. If a slave is using a tristate driver for DTACK*, it can enable its driver upon selection. It may not drive DTACK* low until 30 ns after DS[1..0] assertion and *must* drive DTACK* high within 25 ns of all strobes high (AS*, DS0*, and DS1* = 1). DTACK* must be tristated within 50 ns of all strobes becoming high—20 ns before the next selected slave is permitted to drive DTACK* low.

***The SSBLT revision to the
VMEbus standard provides the
timing rules for use of a
rescinding DTACK* driver.***

VMEbus protocol does not allow multiple slave transactions that might result in slaves with three-state DTACK* drivers attempting to drive DTACK* to opposite levels. The timing rules provide a period of time in which both the previous and newly selected slaves may drive DTACK* high; however, this is not a problem. The only possibility for compatibility problems due to use of a three-state driver for DTACK* exists when the slave is participating in a proprietary broadcast scheme. In such a case, the slave's DTACK* driver can and should be controlled so as to emulate an open-collector output.

The VMEbus standard already specifies a high-current, three-state driver for DS0*; SSBLT adds that requirement for DTACK* since DTACK* must function like DS0* for SSBLT read cycles. Standard 48-mA drivers support data lines, while 64-mA drivers support DS0*, DTACK*, and other control signals.

Transfer length, burst termination

The SSBLT transfer mechanism permits from one to 256 transfers in one block, based upon the requirement that the burst ends at the first 2-Kbyte boundary. The burst can continue only after another address phase, which appears on the VMEbus as a second SSBLT. This arrangement limits the size of the address counter required at the slave and means that boards that are not involved in a transfer don't have to increment their address counters during it.

The master terminates an SSBLT by driving AS* high. For both reads and writes, if AS* changes to high before data capture time, data cannot transfer, and the cycle ends. By sampling AS* at what would have been the data capture time,

the slave can determine that the burst has ended. If AS* is high, the cycle ends, and no data transfer becomes associated with the previous strobe edge. To keep the bus timer enabled and thus prevent specious error indications, DS1* is asserted throughout a burst.

Throttling

The SSBLT provides interblock throttling as a packet-level mechanism corresponding to cycle-by-cycle handshaking. Ideally, an SSBLT slave asserts DTACK* during an address phase; it signals its ability to accept/provide a burst at the full transfer rate. Subsequently, it needs to throttle only infrequently and momentarily. An intrablock throttling mechanism answers this need.

Some applications, such as digital imaging systems, involve large block transfers. It can be necessary to suspend these momentarily to allow a competing transfer to take place on the local bus of the master or slave. This is an example of an appropriate use of *intrablock* throttling.

To delay another block transfer, the destination (slave) can make use of two options. During the address phase, it can simply fail to respond until it is ready, or it can assert both RETRY* and BERR*. The source (master) then terminates the cycle before any data transfers, releases the bus, and waits before attempting another transfer. This is *interblock* throttling. Note that the RETRY* protocol specifies a bus release that usually results in other VMEbus traffic before the retry takes place.

SSBLT also provides a method for intrablock throttling. This alternative activates when the destination's input buffer is almost full, yet the burst is not over. Intrablock throttling allows the destination to suspend the data transfer until it can catch up. System designers should arrange that intrablock throttling is required only infrequently.

To employ intrablock throttling during a write, the slave drives DTACK* to a high level. When the master detects DTACK* as high, it simply suspends the transfer until DTACK* becomes low again. Figure 5 shows intrablock throttling for the slowest case in which the master doesn't suspend transfers until it has driven the third data and strobe edges after DTACK* deassertion. A faster responding source might also have paused with either D[63..0](4) or D[63..0](5) valid on the bus; destination devices must be able to deal with any of these possibilities.

During a read, the master temporarily stops the slave from transmitting data by driving DS0* high. When the master drives DS0* low again, the slave continues the transfer. Figure 6 shows the waveforms for intrablock throttling on a read. The slave's response to the deassertion of DS0* on a read is analogous to the master's response to DTACK* on a write. The same possible stopping points of one, two, or three transfers past strobe deassertion exist as in the write case.

In intrablock throttling for both reads and writes, the source

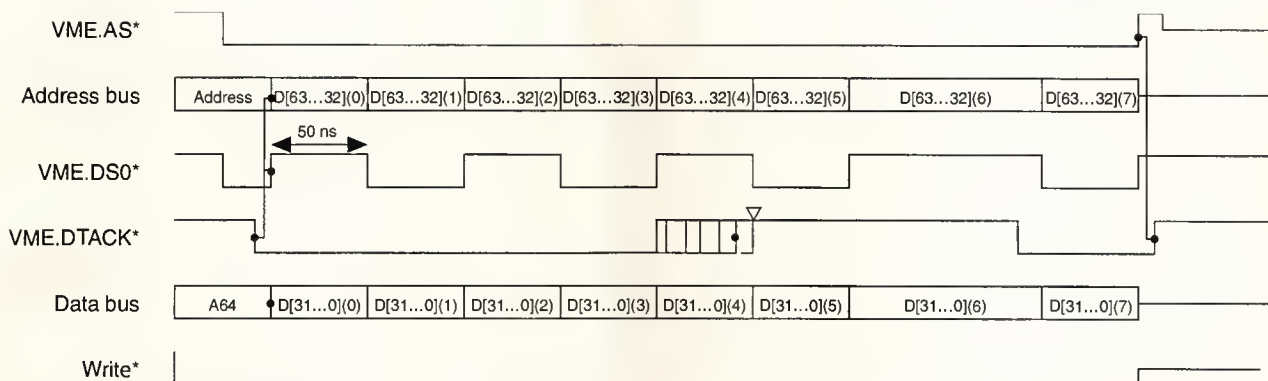


Figure 5. Write intrablock throttling. Throttling protocol rule: Source must freeze its data and clock output within 100 ns of detecting DTACK* high. Since a 30-ns round-trip path delay may exist between source and destination, the destination should throttle when its queue is within 3 locations of the overflow point.

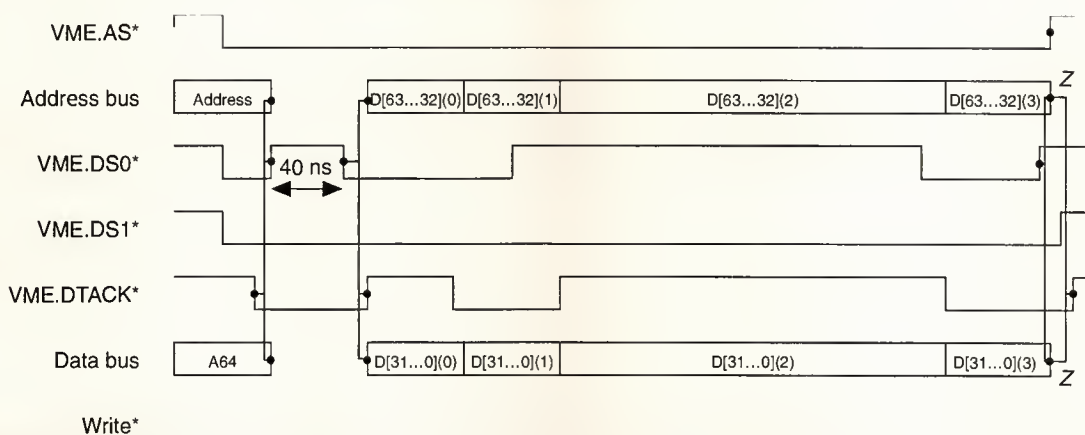


Figure 6. Read intrablock throttling.

must freeze its data and clock outputs within 100 ns of detecting the throttle signal. Note, though, that the timing for the deassertion of DTACK* or DS0* is not fixed; no specific time reference drives either signal. Similarly, no specific time reference determines when the sender of the data must sample DTACK* or DS0*. The SSBLT specification only requires the data sender to halt data transfers within 100 ns of detecting either signal as high.

In Figure 7 (on the next page) two unspecified timings relate to throttling and indicate the added timing consequences of backplane delay. The timing values in this figure also apply to error handling (more on this later).

After asserting the intrablock throttling signal, the destination must be able to accept as many as two additional transfers before the source stops transferring data. Because the round-trip backplane delay between the source and destination might be as great as 30 ns, the destination should throttle

bursts when its queue is within three transfers of the overflow point.

Throttling can be misused as a slow form of cycle-by-cycle handshaking. VMEbus Rev. D will recommend that interface designs not only use throttling infrequently but also only for short periods. The spirit of the revision's SSBLT protocol is that data be burst over the bus at 20 MHz with suspensions only for infrequent, exceptional conditions such as needed for a DRAM refresh.

Conservative timing

The timing for SSBLT is based upon the same backplane and driver characteristics as the original VMEbus specification. This standard provided reliable data transfer with up to 25 ns of skew between data and the data strobe or DTACK*. This skew time includes two 7.5-ns backplane propagation delays plus 10 ns of driver/receiver skew. The two back-

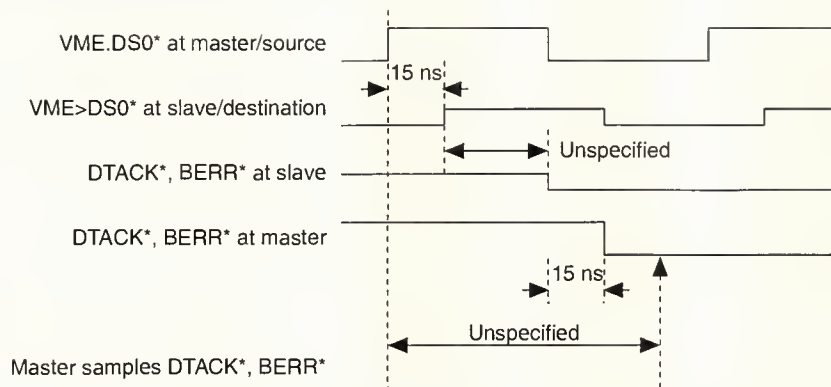


Figure 7. Error and throttling timing. The master must terminate or suspend transfers within 100 ns of detecting a low on BERR* or a high on DTACK*. The points at which the slave asserts BERR* or deasserts DTACK*, or at which the master samples them, are not specified. Note that RETRY* assertion is permitted only on the address phase.

Table 1. Transmission timings.

Characteristic	Timing (ns)
Data clock skew (two backplane propagation delays)	15.0
Data settling time	10.0
Driver/receiver skew	10.0
Receiver setup time	2.5
Minimum capture delay	37.5
Time quantization error (half period of ASIC clock)	5.0
Minimum hold time	2.5
Minimum transmit period	45.0

plane propagation delays appear in the potential skew because the control signal edges (which are driven with 64-mA drivers) might be seen with incident wave switching. The data edges will generally be detected only after their first reflection from the far end of the backplane.

The worst-case minimums for source synchronous capture delay and the overall transmission period take into account the same skew, settling times, setup/hold times, plus a time quantization error allowing this delay to be generated synchronously. The transmission period determines the minimum time that can be allowed between data strobe edges. Table 1 lists these times, in nanoseconds.

To provide an extra margin, the SSBLT protocol adds an

extra 5 ns to the minimum transmission period for a specified transmission period of 50 ns. See Figure 4 again for the SSBLT protocol delays and stable data window.

AM codes

SSBLT employs two new address modifier codes that were previously undefined in the original VMEbus standard. They are 0x06 for A64 SSBLT and 0x07 for A32 SSBLT.

Terminating a transfer

Masters can terminate transfers when the required data has been sent or received. During a write, when the master has transmitted the required data, the master stops strobing the DS0* line and drives AS*, DS1*, and DS0* high. The slave responds by driving DTACK*

high and thus terminating the transfer.

In a read, when the master does not need more data, it terminates the burst by driving DS0*, DS1*, and AS* high. The slave then terminates the transfer within two strobe edges. Figure 2 (shown earlier) illustrates normal terminations for both the read and write cycles. The delays between master and slave result in a "dummy" data cycle being driven onto the bus by the slave after the master terminates the read. The master keeps its AS* signal asserted past the end of the dummy cycle to avoid driver conflicts with the next master.

Either masters or slaves can terminate transfers after detecting an error. If a slave detects an error of any kind during a write, it terminates the transfer by asserting BERR* low. The master then terminates the transfer within two strobe edges. Figure 8 displays a write-error termination.

If a master detects an error of any kind during a read, it terminates the burst by driving DS0*, DS1*, and AS* high. If the slave detects any errors, it also terminates the read by ceasing to toggle DTACK* and asserting BERR* low. In the latter case, the master aborts the transfer. Figure 9 illustrates read-error termination.

VMEBUS WAS ORIGINALLY CONCEIVED as a combination processor-memory-I/O bus.³ Since its inception, processor speeds have increased by a factor of over 100, forcing architects to remove most CPU-memory traffic from the bus. Despite this, the demand for higher bus speeds continues to increase because of the need to support interprocessor com-

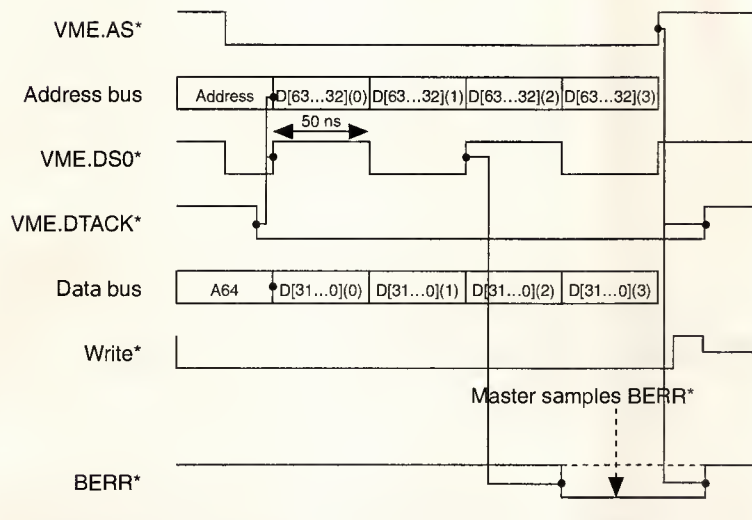


Figure 8. Write with BERR* termination.

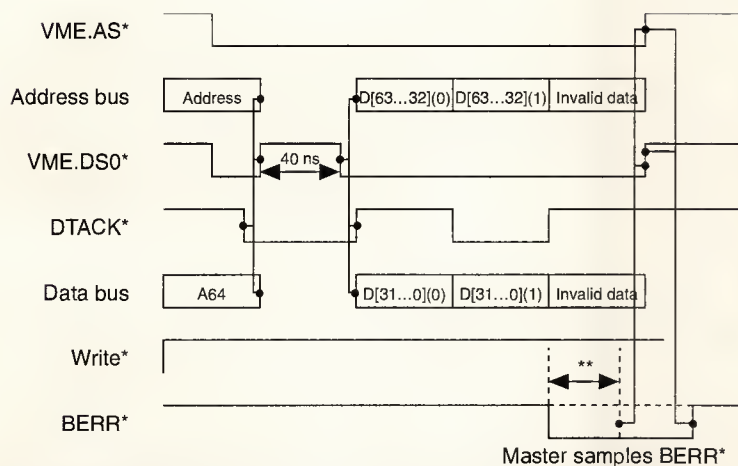


Figure 9. Read with BERR* termination. ** indicates unspecified timing.

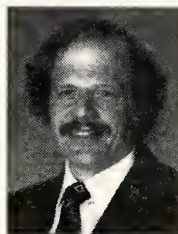
munications at increasing rates and to support higher performance I/O for imaging and graphics, mass storage interfaces, and network communications.

The required bus performance is not a simple function of processor speed. Rather, it is strongly dependent on system architecture and application. The decision to use a particular processor and a particular backplane bus for a particular application incorporates many components other than bus performance. Preeminent among these are cost/performance and risk management. By doubling performance with only a marginal increase in cost, SSBLT doubles the performance/

cost ratio of the VMEbus, widening its market and extending its life quite significantly. By adding performance headroom to the dominant 32-bit and now 64-bit backplane bus standard, SSBLT provides increased assurance that the VMEbus will continue to meet the needs of its users. ■

References

1. IEEE/ANSI Standard 1014, *Versatile Backplane Bus: VMEbus*, IEEE Service Center, Piscataway, N.J., 1987.
2. R.V. Balakrishnan, "The Proposed IEEE 896 Futurebus—A Solution to the Bus Driving Problem," *IEEE Micro*, Vol. 4, No. 4, Aug. 1984, pp. 23-27.
3. D.B. Gustavson, "Computer Buses—A Tutorial," *IEEE Micro*, Vol. 4, No. 4, Aug. 1984, pp. 7-22.



Jack Regula is manager of hardware engineering at Force Computers in Campbell, California. Earlier, he cofounded and served as vice president of research and development of the VME company, Ironics, Inc. He also served as technical chair of the VTC Consortium that designed and developed the VIC chip (VMEbus interface chip). He holds BSEE and MSEE degrees in electronics engineering from Rensselaer Polytechnic Institute in New York.

Direct questions about this article to the author at Force Computers, Inc., 3165 Winchester Blvd., Campbell, CA 95008-6557.

Reader Interest Survey

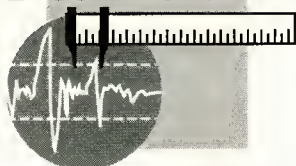
Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 162

Medium 163

High 164

Micro Standards



PCMCIA: The other interface

Recently, Dante Del Corso, our editor-in-chief, sent me a note suggesting that there was a growing interest in PCMCIA, a 68-pin interface for memory cards used in notebook computers. (PCMCIA is the Personal Computer Memory Card International Association.) Dante felt that, although it isn't currently covered by any IEEE or ANSI organization, the de facto standard is having a major impact on the industry.

Agreeing with Dante, is I. Dal Allan, principal consultant at ENDL (Saratoga, California) and a recognized industry expert on interfaces. Allan concedes that the interface has grown from a convenient method of adding slim (3.3-mm thick) memory cards to notebook computers.

Interest in the 68-pin interface seems to be growing simply due to the critical mass of vendors developing notebook and laptop computers. But memory cards aren't the only thing PCMCIA supports. Designers are preparing to use the interface for disk drives as well. Moreover, with emerging 1.8-in. winchester disk drives designers see a good market for add-ins for the portable computers. The emerging interface promises to provide superior interchangeability and better reliability over the long haul. It is rated at more than 10,000 insertions and ensures compatibility over multiple vendors.

PCMCIA defines three types of interfaces. Type I defines the interface for the 3.3-mm memory card.

Type II allows the specification to accommodate storage devices that can't fit the 3.3-mm height constraint. Though the typical height is 5 mm, Type II maintains compatibility to the base standard by using a 3-mm-wide rail along the edges and a 10-mm-deep mating area, both of which are kept at the standard 3.3 mm. The

upshot is that designers won't have to rework slots or cases to manage the larger card.

Still in the proposal stage is Type III, which is supposed to define the interconnection scheme for LANs (local area networks) and modem cards. This definition describes a 50-mm body extension and an 11-mm height. This cavity size seems large enough for the 1.8-in. disk drive form factors. The driving factor is the height since manufacturers want to stay within the 0.5-in. thickness for notebooks. Palmtop computers, though, may pose a different set of problems.

If you are looking for a quick solution and availability for PCMCIA extended products, don't hold your breath. Members of the committee are still wrestling with sizing. For example, three Type I cards can't fit into the Type III cavity, and that is some concern. Additionally, pin size and orientation haven't been worked out.

Among the other issues facing the PCMCIA specification writers are the number of insertions. Although the specification claims more than 10,000 insertions, it is unclear what the real number is. It may be necessary to devise a new seating and release system to minimize wear and ensure proper electrical contacts. Furthermore, PCMCIA has problems with the disk storage capacity for small drives. Consequently, some people talk about providing compression as part of the basic input/output system (BIOS). More than likely, PCMCIA vendors of storage devices will provide a fully integrated card including drive and BIOS with compression capability built in. No doubt Microsoft Corp. (Bellevue, Washington) will want to get its two cents in with a ROM version of its DOS (disk operating system). Whether the Type III cavity can accommodate a full-featured card remains to be seen.

Industry observers such as ENDL's Allan and

Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 7-1230

warren@ssdunx.mdc.com

Richard Steincross from RMS Labs (Long Beach, California) wonder about the cost when compared to other alternatives. Allan points out that the X3T9 group has discussed porting SCSI (the small computer systems interface) to the PCMCIA world but to date has found no takers. The solution seems valid, and protocol chips exist that would support virtually any peripheral.

Even with lack of support from the SCSI community for PCMCIA, Milpitas, California-based Adaptec Inc. is considering a version of its 8000-series integrated disk controller. "That may help on the cost angle," suggests Steincross. He, however, expressed surprise that the emphasis isn't on the IDE (integrated drive electronics) specification. Steincross explains that IDE caught on quickly because it "... was cheap, easy to integrate and heavily supported by the industry. I don't see PCMCIA enjoying as much interest."

Though the jury may not be completely in on PCMCIA, proponents point out the industry infrastructure is growing. Getting on the PCMCIA bandwagon isn't necessarily inexpensive however. Executive and associate memberships carry fees of \$10,000 and \$2,500 a year. An executive membership buys you nine board seats, while an associate is allowed five seats. You can sign up as an affiliate, which allows you to attend meetings, observe, and receive documentation but not participate in discussions. If you are interested in membership or obtaining the latest document, call (408) 720-0107.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 180 Medium 181 High 182

THE TEST ACCESS PORT AND BOUNDARY-SCAN ARCHITECTURE

***edited by Colin M. Maunder and
Rodham E. Tulloss***

The text discusses the approaches for design-for-testability between companies, computers, and the IEEE Standard 1149.1. The tutorial begins with a description of the circumstances that led to the development of the standard, introduces boundary-scan techniques, and provides solutions to the problems faced by this technology. It also discusses the application and implementation of IEEE Standard 1149.1 to the testing and loading of boards and its application to problem areas. Other key topics examined include:

- * The Structure of a Typical Board Test Program
- * Testing and Diagnosis of Standardized Test Logic
- * Silicon Implementation and Related Costs
- * Interfacing to Scan Design and Built-In Self-Test
- * Analog and Mixed-Signal Applications
- * Application to Systems Debugging and Emulation
- * Testing Through the Assembly Hierarchy.

400 PAGES. SEPTEMBER 1990. HARDBOUND. ILLUSTRATIONS.
ISBN 0-8186-9070-4. CATALOG # 2070 \$55.00 / MEMBERS \$44.00

TO ORDER :

Call Toll-free : 1-800-CS-BOOKS

or FAX : 714-821-4010

in California call : 714-821-8380

IEEE COMPUTER SOCIETY PRESS

Customer Service Center

10662 Los Vaqueros Circle

P.O. Box 3014, Dept. 102-3

Los Alamitos, CA 90720-1264

(ADD \$5.00 TO COVER HANDLING CHARGES)

(PRICES ARE 20% HIGHER OUTSIDE THE US AND CANADA)



Richard H. Stern

Oblon, Spivak,
McClelland, Maier &
Neustadt, P.C.

1755 Jefferson Davis
Highway, Suite 400

Arlington, VA 22202

Game Genie: copyrights and add-ons

Users often modify computer programs to enhance their utility. Sometimes they buy add-on programs to increase performance. These add-ons provide interfaces that are easier to learn and remember, increase speed, add new functions, perform additional tasks, and otherwise interact with a preexisting program to provide additional utility. Typically, add-on programs do not permanently modify the underlying program, do not make a tangible copy of a new, modified program, and cannot be used unless the customer has already purchased a copy of the underlying program.

Often, owners of copyrights in underlying programs do not object to add-on programs. The add-ons add to the utility of, and increase the demand for, the underlying programs. There are many circumstances, however, in which copyright owners might be displeased with—and therefore want to suppress—an add-on program.

Consider the case of a low-power version of a program sold cheaply for the low-price market and a high-power version sold upmarket. What if an add-on cheaply converts the down-market model to perform the tasks of the upmarket model? (See the box on a similar case.) Sometimes, an add-on program shows users the inadequacies of the underlying program by providing improvements that the copyright owner has refused to be bothered to make. That may pave the way for customers to migrate to another product. This appears to have occurred in the case of database management add-on programs, which were eventually followed by competing programs that included the features of the add-on programs, as well as still additional improvements.

I am not aware of any litigation over add-on programs of the types just described. However, a recent decision from the San Francisco federal

trial court comes close. In *Lewis Galoob Toys, Inc. v. Nintendo of America, Inc.*,¹ the court rejected a claim that copyright owners have the exclusive right to determine whether end users may temporarily modify computer program-related works once they are in the users' hands. The underlying work in this case was a video game operated by a computer program. But the same legal conclusions would appear to apply with equal force to any other computer program-related work, such as a spreadsheet, database, or word processing program, in a consumer user's hands.

Background. Nintendo, a major Japanese and American seller of home and arcade video game equipment, owns copyrights in many popular video games. It markets these games in the home video field by selling cartridges that connect into its Nintendo Entertainment System (NES) game consoles. These are small special-purpose microcomputers that provide a video display on a home television set. The cartridges fit into the consoles as cassettes fit into audio tape and videotape players.

A typical video game, such as *Super Mario Brothers*, features a protagonist (Mario) whom the player moves across the display screen. The computer system displays obstacles and enemies which Mario must overcome. The player pushes buttons and manipulates controls to cause Mario to jump over obstacles, evade dangers, kill enemies, and traverse a series of "worlds," at the end of which he may rescue a princess from an ogre. Programmed-in constraints limit Mario's abilities. He can jump only so high or so far. He has only so many missiles to hurl at enemies. His speed is limited. To the extent that the player's abilities are insufficient, given the programmed-in constraints, to overcome the dangers that Mario

faces, he succumbs and dies. After a set number of deaths, the player loses and the game ends.

Galoob markets an accessory, the Game Genie Video Game Enhancer. Game Genie fits between a video game cartridge and the NES console. It modifies electronic signals passing between the console and cartridge by temporarily inserting code segments, or *patches*, into the computer program as it appears in temporary memory (RAM). Game Genie thus allows a user to change the constraints, for example, to make Mario jump higher, run faster, and hurl more missiles at adversaries. It can also allow Mario more deaths before the player loses. Game Genie similarly modifies the play of other NES-compatible video games.

Nintendo contended that Galoob was causing its customers to create derivative-work versions of the video game, in violation of Nintendo's copyright. Section 106(2) of the Copyright Act gives a copyright owner the exclusive right to prepare derivative works based on a copyrighted work. Section 101 of the Copyright Act defines a derivative work as a work based on a preexisting work, such as translation, dramatization, motion picture version, art reproduction, condensation, "or any other form in which a work can be recast, transformed, or adapted."

Nintendo claimed that by modifying the program to change the rules of the game Galoob was making a change in Nintendo's copyrighted work that amounted to preparation of a derivative work. The copyrighted works in a video game, according to Copyright Office practice, include a computer program (literary work) and a visual display (audiovisual work). Since Game Genie changed the program by putting patches into the code, and since that changed the visual display, Galoob caused users to make unauthorized, derivative-work computer programs and displays.

Nintendo also markets (or licenses others to market) devices that modify

its video games in various ways, such as speeding up part of a game or skipping stages. But Nintendo maintained that its copyright gives it the exclusive right to traffic in such modifications. Since Galoob caused its customers to trespass on Nintendo's claimed exclusive right to modify the game play, Nintendo accused Galoob of *contributory infringement*—meaning, contributing to, or causing customers to engage in, copyright infringement.

Galoob denied that the modifications created a derivative work. It also asserted the affirmative defense that personal game modification by end users for their personal enjoyment of games they had purchased was a fair use and was therefore privileged. The trial court agreed with Galoob on both counts, and an appeal is pending.

Is there a derivative work? Game Genie does not make a physical copy of the computer code stored in a Nintendo cartridge. Its electronics and code patches merely interact with those of the NES console and the game cartridge, modifying signals to change the video display and the results of game action.

For example, a user might set Game Genie to continue the game until Mario dies six times, rather than three. In effect, Game Genie substitutes its instructions and data for parts of the original program. (For example, "do so-and-so

for $i = 1$ to 3" becomes "do so-and-so for $i = 1$ to 6.")

But this change occurs only temporarily, without rewriting the ROM in the Nintendo cartridge. Game Genie is like Maxwell's Demon. It sits between the copyrighted computer program in the cartridge and the NES console's CPU and censors the messages that go back and forth. Since it does not write anything down in a fixed form, it does not make a tangible, more-than-transitory copy of any of Nintendo's computer program or visual display. The modified code exists only in RAM, and the modified display appears only temporarily on the TV screen.

However, section 106(2) of the Copyright Act does not require that one make a permanent copy. It gives copyright owners an exclusive right to prepare derivative works in tangible or intangible form. Thus a stage performance of the musical *Cats* without authorization from T.S. Eliot would infringe the copyright in his book of poems, *Old Possum's Book of Practical Cats*, even if no written script was reproduced. Therefore, Game Genie's program modifications apparently prepare a derivative work, in terms of both the program and the audiovisual work whose display the program causes.

Moreover, one federal appellate court has already held very similar conduct to be infringing preparation of a

Third-party upgrade

In *Hubco Data Prods. Co. v. Management Assistance, Inc.*, 219 USPQ 450 (D. Idaho 1983), Hubco, the copyright owner, sold different versions of its computer program designed to serve computer systems having different amounts of memory. Hubco charged a higher price as the amount of memory handled increased. Hubco also sold upgrade services. MAI, the infringer, engaged

in a competing upgrade service (not the sale of an add-on program as such), by modifying the code of installed Hubco programs to make them serve more memory. The court based its decision on legal theory that MAI infringed by reproducing copies of the copyrighted program in the course of decompilation and study undertaken to learn how to make appropriate upgrades.

derivative work. In *Midway Manufacturing Co. v. Artic International*,² the court found that use of speedup kits to change the operation of arcade video games violated section 106(2). The speedup kit made the game harder to play, to be sure, while the Game Genie makes the game easier, but that merely reflects different user purposes being served.

In the speedup case, arcade proprietors wanted to make the game harder because customers found it so easy that they either lost interest or lingered over it for an interminable time without inserting additional coins. The speedup kit therefore improved the revenue that arcade owners could earn from video game equipment they had purchased to earn revenue.

In the present case, some users find the game too hard to enjoy playing. They therefore want to improve their enjoyment from the game which they (or their parent) purchased to provide them with home entertainment. Whether the modification makes the game harder or easier does not meaningfully change the fact pattern of the Artic case from that of the Galoob case. The key fact is that the alleged infringer's conduct alters the code and display.

No fixed copy. The Galoob court made a point of the lack of a tangible copy of the modified game program. It noted that the modified version would not be transferrable to a third person because there was no copy. But that would be relevant only if some other section—not section 106(2)—were involved. Section 106(1) prohibits making unauthorized copies. Section 106(3) prohibits transferring unauthorized copies to others. But this case did not allege a copyright infringement under those sections of the Copyright Act.

The court sought to support its ruling by the statutory wording of the definition of derivative work, which includes the phrase "or any other form in which a work may be recast, transformed, or adapted." It said that "form"

implied fixed and tangible form. The court was not made aware of the legislative history of the definition, which is contrary to the court's theory.

***The statute is a
mess. But it is not
the trial court's
job to rewrite it.***

The House Report accompanying the 1976 Copyright Act points out that the omission in section 106(2) of any requirement of fixation in a tangible form was intentional. Section 106's anti-reproduction and antidistribution clauses contain fixation-in-copy requirements, but that requirement was deliberately omitted from section 106(2)'s provision against preparation of a derivative work—albeit for rather frivolous reasons.

The report explained that the forms of some copyrightable works lend themselves to preparation of derivative works in impermanent or intangible form. Yet they deserved protection against unauthorized takings, which should therefore be defined as infringements. Congress cited pantomime and ballet as examples illustrating the claimed need to eliminate the fixed-copy requirement for infringement by making derivative works. To save pantomime from piratical derivative works, therefore, Congress made preparation of derivative-work versions of pantomimes—and all other works—a copyright infringement, regardless of whether a tangible copy was made. Indeed, Congress did not even require as a condition of infringement liability that anything be done with the unauthorized derivative work.

Congress may have made an unwise or even foolish decision. It should have

limited liability for preparing derivative works in intangible form to pantomimes and similar works, so the statutory remedy would not sweep up conduct unrelated to its legislative concern. At least Congress should have required some kind of use after preparation before liability attached. The statute is a mess. But that does not mean that the trial court should rewrite it to correct the legislative error. That is not its job under our legal system.

Users' rights. In further support of its construction of the phrase "derivative work," the Galoob court pointed to the nature of the competing interests at stake. It said the copyright law's purpose is to balance "a fair return on an author's creative labor against the need for 'broad public availability of literature, music, and the arts.'" Galoob sells Game Genie to users who have already paid Nintendo its price for the video game cartridge. Users modify the games only for personal enjoyment, not for commercial gain. The conduct is analogous to skipping commercials on a videotape of a television program by fast-forwarding, or rewinding and viewing in slow motion a critical play of a football game. None of this, the court said, deprives the copyright owner of the opportunity to derive "current or expected revenue."

(The facts are somewhat more complicated than the court paints them, although on balance its assertion may accurately characterize them. The court did not mention here that Nintendo sought to gain added revenue from its customers by marketing somewhat different game modification devices to them. One could argue, therefore, that to whatever extent Galoob satisfies this market, Nintendo cannot instead satisfy it for its own profit and is therefore deprived of expected revenue. In response, Galoob might make two points. First, to date Nintendo has neglected the needs of these particular customers and left them unsatisfied or else they would not be Game Genie

customers. Second, why should Nintendo have a monopoly over this ancillary market? That is something to be decided, not assumed.)

The court summed up the equities of the case as follows: "Having paid Nintendo a fair return, the consumer may experiment with the product and create new variations of play, for personal enjoyment, without creating a derivative work For these reasons, this Court finds that the Game Genie does not create a derivative work protected by the copyright laws."

The result may be correct, but the legal analysis has gaps. The court's argument is sound for creation of a defense of estoppel, privilege, or implied or constructive license. But such an affirmative defense is quite distinct from the proper statutory construction of the phrase "derivative work."

Fair use. As an alternative holding, and assuming in the course of the argument that Game Genie prepared a derivative work, the court went on to find a fair use. Fair use is a statutory privilege codifying a series of judicial decisions favoring certain uses of a copyrighted work as privileged or immunized from liability. The privilege is the end user's in the first instance. But it extends to a person charged with contributory infringement liability because of responsibility for an end user's conduct—for example, if the person sold the end user the equipment used to commit the alleged copyright infringement. There can be no contributory infringement without direct infringement. Hence, if the end user's alleged direct infringement is privileged as fair use, then the accused contributory infringer cannot be punished in damages for contributing to a non-existent direct infringement.

That the end user's use is noncommercial creates a rebuttable presumption in favor of fair use. Here, the end user uses Game Genie for private home entertainment. This places the facts of the case on a par with those of *Sony Corp. v. Universal City Studios, Inc.*³ In

that case, the Supreme Court found it fair use for users of home videotape recorders to record broadcasts for later viewing.

***If the user's
conduct is
privileged as fair
use, the supplier
cannot be
punished for
contributing to a
nonexistent
infringement.***

Another factor in favor of a finding of Game Genie fair use, the court concluded, was that the end users had already paid Nintendo for the video game cartridge. The court felt that purchase of the game cartridges gave customers a right to maximize their enjoyment of their purchase, including by modifying how the product worked.

Finally, the most important factor in the fair-use analysis was that Nintendo could not show that Game Genie would adversely affect the market for the copyrighted work by diverting sales away from Nintendo. The court rejected Nintendo's claim that Game Genie harmed the "Nintendo Culture," a concept the company promoted as "the apex of Nintendo's marketing strategy ... a [customer] mind-set intentionally created by Nintendo." Part of the Nintendo Culture, according to its marketing experts, is peer rivalry among video game players, who gain prestige by achieving high scores in the game. Players verify their achievement by

photographing a screen displaying the high score. If everyone could get high scores with Game Genie, Nintendo said, then "this socially reinforcing practice would fall by the wayside," and Nintendo would lose future sales.

The court refused to believe this theory because Nintendo was itself marketing products and a magazine that helped players modify game play in a manner similar to Game Genie. In any event, the court would not award Nintendo in litigation what Nintendo could not achieve by its competitive efforts in the marketplace—"the exclusive right to modify game play as it alone sees fit"—because the Copyright Act does not bestow that power on copyright owners.

Implications for add-on programs. Much of what the court said carries over from patching video game computer programs with a Game Genie to patching an application with an add-on program. First, the modified code exists temporarily in RAM and is not written into ROM. That, of course, disregards the peculiar history of the part of the Copyright Act dealing with copyright infringement by preparation of derivative works.

More important, the modification occurs for the benefit of an end user who has already purchased a copy of the underlying copyrighted computer program. By the same token, the copyright owner has already been paid once for the right to use the program. The end use may be for commercial or noncommercial purposes, depending on the user. No one sells or transfers the modified program to others. Finally, one can assert that use of the add-on program will not divert sales away from the copyright owner.

These factors, on balance, suggest that the verdict in an add-on copyright infringement suit should be against the copyright owner and in favor of end-user rights. But the chain of argument summarized above has defects which might lead a different court to reach a different result. On the other hand, the

Galoob court left unmentioned other arguments favoring add-ons and end users' rights which, if properly presented, might lead another court to the same result by another route.

Alternative analyses. The Galoob court's opinion is a dog's breakfast. First, a derivative work was probably prepared, even though it was not a fixed, tangible copy. The existence of such a copy indicates the need to consider possible affirmative defenses (fair use is only one) that may negative the case of copyright infringement. Even when a copyright is infringed, sometimes circumstances excuse the infringement.

There may well have been a fair use, but the court's legal analysis of fair use is flawed by its overstatement of the case. Such overstatement is inherent in fair-use analysis. The legal standard for determining whether a use is fair calls for a balance of four incommensurable factors, in the form, fair use = apples + oranges + lemons + grapes. The only way the court felt it could balance the factors against one another was to say that none of them favored the losing party. Otherwise, the court would have been compelled to decide whether the apples carried more legal weight than the oranges—a daunting task.

(By way of analogy, to find a minimum or maximum of $F(x,y,z,t)$, you must solve for the partial derivatives of F with respect to x , y , z , and t each successively being set to zero. Otherwise, you cannot tell that you have a peak rather than merely a point along a ridge or a saddle point.)

The court defined away the problem by assuming that Game Genie sales were not diverting sales of Nintendo's copyrighted product. That tipped the most important of the fair-use factors wholly in Galoob's favor. But Nintendo was, by the court's own account, merchandising a set of ways to prepare derivative-work versions of the copyrighted video games, while Game Genie provided another. Arguably, this resulted in competing versions of the

copyrighted work in the same marketplace. That fact, if it is one, casts doubt on the correctness of the court's conclusion that the alleged copyright infringement did not supplant Nintendo to any degree as a seller of the copyrighted work.

An add-on program can dry up the market for later versions.

The same kind of thing can occur with an add-on program. Consider 4DOS, a shareware add-on program that interacts with MS-DOS. 4DOS, which has been available for several years, offers its users many functions that Microsoft did not include in MS-DOS 2 and 3—for example, on-line help with DOS command meanings and ability to recall and edit prior command entries (by using arrow keys). I am in no hurry to upgrade to MS-DOS 5, since 4DOS already provides most of what I would get from it (and most or all of the rest is found in old, already-installed programs such as 1Dir+ and PC Tools). An add-on program can thus dry up the market for later versions of the underlying program.

That does not necessarily tip the fair-use analysis in Nintendo's favor or against add-on programs in general. But the need to attempt an apples versus oranges fair-use analysis, instead of defining it away, makes the fair-use approach much more precarious. An alternative legal analysis may therefore be preferable, if it supports the same result.

The court's repeated emphasis on user rights points the way toward several possible such analyses. One is the legal doctrine of implied license. In

patent law, a purchaser of a patented product has an implied license to modify it to increase its value. For example, the purchaser of a canning machine may modify it to process a different size can. The implication of the license is apparently by action of law, not from the surrounding facts. That is, the court implies the license based on its sense of fitness, not on the basis that the parties actually intended to agree to a license. That suggests that an attempted disclaimer of the implied license by the patent owner would be ineffective. The argument from patent law has not yet been carried over to copyright law. It may fail, but it is worth considering. Implied license would seem to be at least as strong an argument as fair use.

A related alternative legal argument is estoppel: The seller is estopped from preventing the customer from fully enjoying the use of purchase. There appears to be no substantial difference between estoppel and implied license. Estoppel is just another name for the idea that, by selling the product to the customer, the seller has acted in a manner inconsistent with preventing the customer from fully benefiting from the sale.

The doctrine against derogation from grants provides yet another legal argument amounting to the same thing. In *British Leyland Motor Corp. v. Armstrong Patents Co.*,⁴ the House of Lords held that a car manufacturer, who owned copyrights covering tail pipes and other spare parts, could not require car owners to procure spare parts only from its licensees. The court considered that to use copyright law for this purpose would derogate from the title to the car that the manufacturer had conveyed to the customers upon sale. Any added expense or inconvenience imposed on car purchasers that interfered with their enjoyment of the purchased goods would derogate from the grant of title. Therefore, the court would not permit the seller to assert its copyrights to cause such results.

Finally, probably the strongest alter-

native legal argument, one wholly unmentioned by the Galoob court, could be based on section 117 of the Copyright Act. Section 117 gives owners of copies of computer programs a right to make adaptations of the programs when they do so as "an essential step" in their utilization of the computer programs. The NES console is a computer, within any reasonable definition of that term. It is a low-end, special-purpose microcomputer having a microprocessor chip as central processing unit. The video game cartridge contains a stored computer program, among other things. At least *prima facie*, the fact situation is that defined by section 117.

The only problem areas are

- Does the fact that modifying the computer program also modifies an audiovisual work take the case outside section 117?
- Is the adaptation "an essential step" in utilization of the computer program?

On the first point, a court would probably regard the program and audiovisual display as a unitary work. That is how the Copyright Office registers them. Therefore, the right to modify the program should carry with it the right to let the modifications change the audiovisual display since the two are a single legal unit.

The second point is less predictable. Legal authority is divided, but the better view is that "an essential step" is one intrinsic to the contemplated use of the program or one that the end user strongly desires to accomplish.⁵

These alternative rationales for the Galoob court's result, singly or in combination, would probably have given stronger support to the judgment than the fair-use analysis did. They are not different in kind, however, from the court's rationale of focus on users' rights. Where the proposed alternatives differ from the court's approach is that they seek to provide a legal theory cen-

trally based on users' rights rather than to invoke such rights as incidental support for another legal theory.

One may quarrel with the chain of legal reasoning in the Galoob decision, but Galoob definitely tells us something important. It suggests that courts favor the rights of the customer in an add-on situation, at least when copyright owners cannot show that the customer is taking a free ride at their expense. As Justice Black once said about repair and reconstruction of patented products, "One royalty to one patentee for one sale is enough."⁶

Courts are likely, therefore, to feel that customers deserve considerable freedom to modify a purchased computer program. The sentiments about relative equities expressed in the Galoob decision may thus be more precedential, for prediction purposes, than the legal analysis.

References

1. 20 USPQ 2d 1662 (N.D. Cal. 1991) (12 July 91).
2. 704 F. 2d 1009 (7th Cir.), cert. denied, 464 US 823 (1983).
3. 464 US 417 (1984).
4. [1986] 1 All Eng. Rep. 852 (H.L.).
5. Foresight Resources Corp. v. Pfortmiller, 13 USPQ 2d 1721 (D. Kan. 1989).
6. Aro Mfg. Co. v. Convertible Top Replacement Co., 365 US 336, 360 (1961) (concurring opinion).

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 177 Medium 178 High 179

Micro Review

continued from p. 7

I installed Disk Express II and watched it go through its paces. I don't think my hard disk was badly fragmented, so I'm not sure how much improvement I've seen, but I am sure of two things. Disk Express II is easy to use, and I feel a lot better knowing that its many features will be available if odd situations arise.

Multidisk is a hard-disk partitioning program. It allows you to place files into logical groups kept together on your disk to minimize access times. A number of protection features provide advantages on networked systems or on systems that more than one person can access.

Alsoft also claims that partitioning your disk provides another layer of protection against computer viruses. I have not had virus problems on my Mac, but my PC was recently infested by the notorious Michelangelo virus. That's a story for another column.

Disk Check diagnoses disk and directory problems. It couldn't find any problems with my disk or directories, but it did help me identify and remove an invisible anchored file belonging to a program I got rid of long ago.

If you use your Macintosh regularly, Disk Express II will certainly improve your efficiency. I think it's worth the modest price.

I don't have space to describe all of the other interesting programs I've received recently. Some of these will appear in future columns.

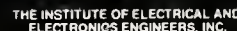
Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 183 Medium 184 High 185



Membership/Subscription Application



COMPUTER

Scientific Visualization
Research and Development

Computer's popular departments include New Products, Product Reviews, Book Reviews, Standards, Open Channel (a reader forum), and Computer Society News.

- Discounts of up to 50% on **Computer Society Press books and videos**. Over 700 titles covering a broad spectrum of computer science topics including networking, communications, advanced systems, image processing, security, artificial intelligence, and visualization
- Discounts on registration to over 100 conferences, workshops, and tutorials each year
- Low subscription rates on special-interest periodicals
- Opportunity to participate in any of our 32 technical committees—networks of professionals with common interests in computer hardware, software, and applications
- Opportunity to participate in the development of more than 200 standards projects sponsored by the society
- Low member rates on COMPMAIL, the electronic mail service especially designed for computer professionals

[illegible]

Mo.		Yr.	

Exp. Date

Checks drawn in local currency on a bank in the country of origin accepted from members in Australia, Austria, Belgium, Canada, Denmark, France, Germany, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland, UK, USA, and Yugoslavia.

To join: see item 1, 2, or 3.
To subscribe: see item 4.

	Half Year	Full Year
	Mar 1-Aug 31	Sept 1-Feb 28
PRICES EXPIRE 12/31/92		

- | | | |
|--|----------------------------------|--------------------------------|
| I reside in Region 1-6 (United States)..... | <input type="checkbox"/> \$58.50 | <input type="checkbox"/> \$117 |
| I reside in Region 7 (Canada) | <input type="checkbox"/> \$53.50 | <input type="checkbox"/> \$107 |
| I reside in Region 8 (Europe, Africa, or the Middle East)..... | <input type="checkbox"/> \$53.00 | <input type="checkbox"/> \$106 |
| I reside in Region 9 (Latin America)..... | <input type="checkbox"/> \$49.50 | <input type="checkbox"/> \$ 99 |
| I reside in Region 10 (Asia and Pacific) | <input type="checkbox"/> \$48.50 | <input type="checkbox"/> \$ 97 |

IEEE Member Number _____

	issues per year			
<i>IEEE Computer Graphics and Applications</i>	6	<input type="checkbox"/>	\$12.50	<input type="checkbox"/> \$ 25
<i>IEEE Design and Test</i>	4	<input type="checkbox"/>	\$11.00	<input type="checkbox"/> \$ 22
<i>IEEE Expert</i>	6	<input type="checkbox"/>	\$10.00	<input type="checkbox"/> \$ 20
<i>IEEE Micro</i>	6	<input type="checkbox"/>	\$11.50	<input type="checkbox"/> \$ 23
<i>IEEE Software</i>	6	<input type="checkbox"/>	\$12.50	<input type="checkbox"/> \$ 25
<i>Transactions on:</i>				
<i>Computers</i>	12	<input type="checkbox"/>	\$12.00	<input type="checkbox"/> \$ 24
<i>Knowledge and Data Engineering</i>	6	<input type="checkbox"/>	\$ 9.50	<input type="checkbox"/> \$ 19
<i>Parallel and Distributed Systems</i>	6	<input type="checkbox"/>	\$ 9.50	<input type="checkbox"/> \$ 19
<i>Pattern Analysis and Machine Intelligence</i>	12	<input type="checkbox"/>	\$12.00	<input type="checkbox"/> \$ 24
<i>Software Engineering</i>	12	<input type="checkbox"/>	\$11.00	<input type="checkbox"/> \$ 22
<i>IEEE Annals of the History of Computing</i>	4	<input type="checkbox"/>	\$ 8.00	<input type="checkbox"/> \$ 16

Periodicals total	\$ _____	CANADIAN residents	
CA, DC residents add		add 7% GST; BELGIAN	
applicable sales tax	\$ _____	residents add 6% VAT	\$ _____
Membership fees	\$ _____		
Total	\$ _____	Total enclosed	\$ _____

I hereby make application for Computer Society membership and, if elected, will be governed by IEEE's and the society's constitutions, bylaws, and statements of policies and procedures.

Full signature _____ Date _____
☐ Male ☐ Female

First name	Middle initial(s)	Last name	Date of birth
------------	-------------------	-----------	---------------

Street address	City	State/Country	Zip
----------------	------	---------------	-----

Course	Degrees received	Date
--------	------------------	------

Name of educational institution _____

Title or Position	Name	Address	City	State	Zip	Phone	Fax	E-mail
President	John Doe	123 Main St	New York	NY	10001	(212) 555-1234	(212) 555-5678	john.doe@company.com
Vice President	Jane Smith	456 Elm St	Los Angeles	CA	90001	(213) 555-2345	(213) 555-6789	jane.smith@company.com
Secretary	Bob Johnson	789 Oak St	Chicago	IL	60601	(312) 555-3456	(312) 555-7890	bob.johnson@company.com
Treasurer	Alice Brown	101 Pine St	Houston	TX	77001	(713) 555-4567	(713) 555-8901	alice.brown@company.com
Director	Charlie Davis	202 Maple St	Phoenix	AZ	85001	(602) 555-5678	(602) 555-9012	charlie.davis@company.com
Director	Diana Evans	303 Birch St	San Francisco	CA	94101	(415) 555-6789	(415) 555-0123	diana.evans@company.com
Director	Ethan Green	404 Cedar St	Seattle	WA	98101	(206) 555-7890	(206) 555-1234	ethan.green@company.com
Director	Fiona White	505 Spruce St	Portland	OR	97201	(503) 555-8901	(503) 555-2345	fiona.white@company.com
Director	George Black	606 Ash St	Denver	CO	80201	(303) 555-9012	(303) 555-3456	george.black@company.com
Director	Helen Gray	707 Red St	San Diego	CA	92101	(619) 555-0123	(619) 555-4567	helen.gray@company.com
Director	Ivan King	808 Blue St	Austin	TX	78701	(512) 555-1234	(512) 555-5678	ivan.king@company.com
Director	Julia Lee	909 Yellow St	Nashville	TN	37201	(615) 555-2345	(615) 555-6789	julia.lee@company.com
Director	Kevin Miller	1010 Purple St	Boston	MA	02101	(617) 555-3456	(617) 555-7890	kevin.miller@company.com
Director	Laura Wilson	1111 Green St	San Jose	CA	95101			

Firm name

Firm address

City	State/Country	Zip
------	---------------	-----

ENDORSER (an IEEE member, Senior Member, or Fellow)

Endorser's signature _____

Name (print in full) _____ IEEE Member No. _____

Street address

City	State/Country	Zip
------	---------------	-----

Mail or fax to appropriate Computer Society office:

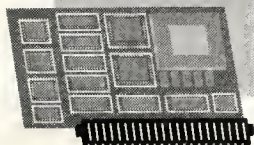
EUROPE: 13, Avenue de l'Aquilon, B-1200, Brussels, BELGIUM. Fax: 32-2-770-8505

PACIFIC RIM: Ooshima Building, 2-19-1 Minami-Aoyama, Minato-ku, Tokyo 107, JAPAN. Fax: 81-3-3408-3553

ALL OTHERS: 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264 USA. Fax: 714-821-4010

MICRO 4/92

On the Edge



Carl Warren

McDonnell Douglas

Space Systems Company

(714) 896-3311

x. 7-1230

warren@ssdunx.mdc.com

Regression testability

[This issue, Lee White and Hareton Leung make an argument for a more systematic approach to regression testing in software development.

I invite readers to send information on a tool or method that solves problems, for consideration in future columns. — C.W.]

Lee White

Case Western Reserve University

Hareton K.N. Leung

Bell-Northern Research

Designers consider many criteria when writing software. In these times when change is so common, one of the most important is *maintainability*, which strongly correlates with other desirable criteria. We propose a more systematic approach to an important aspect of maintainability: regression testing.

We can differentiate between *perfective* maintenance, which enhances software functionality, and *corrective* maintenance, which detects and corrects defects. Whether the maintenance is perfective or corrective, we must ensure that it does not inadvertently affect unmodified functionalities. When this occurs, we call it a *regression error*. Regression testing uses test data previously developed for these unmodified functionalities to detect such regression errors.

Regression testing is important at the unit, integration, and system (or functional) testing levels. However, software development teams usually have responsibility for unit and integration testing. These teams do not consistently apply regression testing at these levels when they make changes and often do not even systematically retain test data. System or functional testers, on the other hand, are very systematic about keeping test data and

applying regression testing. This costs more than detecting regression error earlier.

We recently completed a research project,^{1,2} where for small changes we endeavored to identify subsets of test data to use as regression tests at the unit, integration, and system levels. Our idea was to cut the cost and time to run the regression tests for numerous small changes in the software and to focus on the areas of tests related to code or functionalities of the change.

Figure 1 illustrates the approach for regression testing at the unit level so a subset of the test data can detect a unit regression error. The static analyzer detects those test data, called reusable tests, that cannot be affected by the program changes. We could accomplish this detection with static slices, as proposed by Weiser,³ which identify statements that could be affected by program changes. If this fails to lead to a sufficient reduction in test data, we could use dynamic slicing, as introduced in Korel and Laski,⁴ which indicates statements actually affected by program changes.

The dynamic analyzer executes the remaining tests and identifies obsolete tests that no longer achieve their intended function since the program has changed. The remaining testable tests reveal regression errors. We require new tests to update functional tests if the specification of the module has changed, or to obtain structural tests to achieve a specified level of coverage. To do so, we can use the module dynamic tester shown in Figure 1.

The method we propose may result in a higher payoff situation for regression testing at the integration level. We endeavored to develop a general approach that does not depend on the particular type of integration (such as top-down, bottom-up, or hybrid), as long as it is incremental.

Firewall. A key question is, given the modified modules, how many modules must we retest? In other words, where can we draw a *firewall* around those modules which must be retested? When we detect errors, we should make changes so as to keep the firewall from spreading to include more system modules if possible.

To make this analysis precise, we assume that all module dependencies are indicated in the call graph. Figure 2 shows an example call graph with four modified modules C_1 - C_4 . This is a severe assumption because global variables are another common dependency in many programs, in which a number of modules may define global variables used by otherwise unrelated modules. I will briefly discuss global variable testing later.

Resource dependencies may exist between modules. An example is memory, in which the resource is constrained between a number of modules. We assume to indicate a precise result, the only errors present are due to the modified modules from the maintenance effort. We also assume reliability of the unit and integration tests. (I will return to this assumption at the end of this analysis.)

Given these assumptions, we analyzed a number of basis cases describing module dependencies in the call graph.² The possibilities for a module a are

- no change in a , NoCh(a),
- only code change in a , CodeCh(a), and
- change in the specification of a , SpecCh(a).

If module a calls module b , then we can ignore any cases in which neither a or b is changed. This leaves us with eight cases to consider. We must also model the addition of new modules and deletion of modules for which we have identified several more cases.

Figure 3 on the next page indicates the four critical boundary cases. Two cases correspond to an unchanged module calling a modified module, and

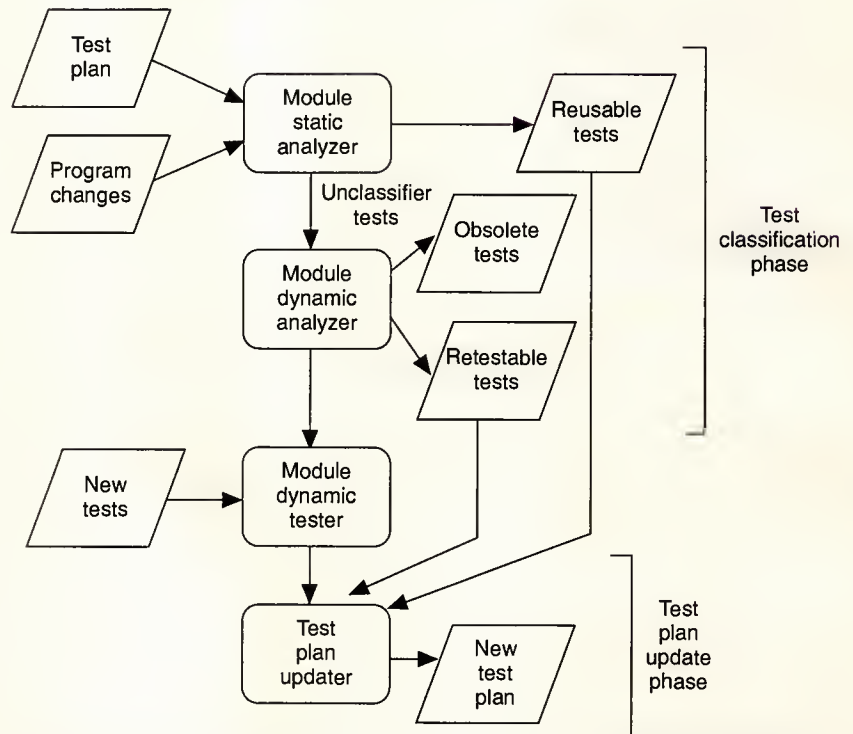


Figure 1. Unit regression testing.

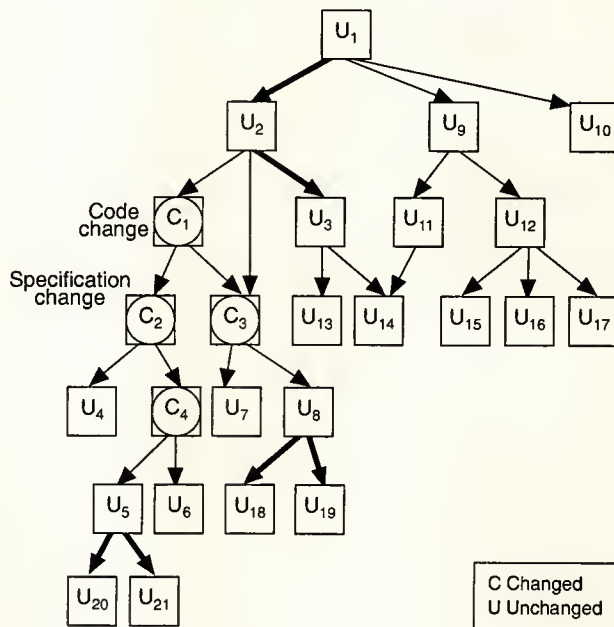


Figure 2. Basis cases and the calculation of a firewall, indicated by bold arrows.

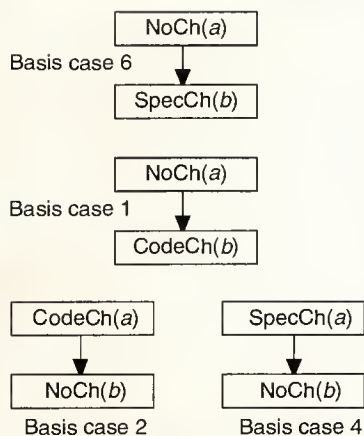


Figure 3. Boundary cases for firewall construction.

the other two correspond to a modified module calling an unmodified module. Case 6 in Figure 3 is unusual in that one would not expect an unchanged module to call a module in which the specification has changed. This unusual situation creates two consequences.

- We should reexecute not only the integration tests between modules but also the unit test of the calling module.
- Case 6 is the only case in which we cannot guarantee that only the modified module needs to be changed if any tests detect an error. If the calling module is modified, the firewall expands.

The other three cases are simpler and can be characterized by the following observations:

- We need to reexecute only the integration tests between the two modules in each case. We do not have to rerun the unit tests for the unchanged module.
- If any of the tests detects an error, we can correct the error by changing only the modified module. The programmer should not change the unmodified module. If we fol-

low this discipline, the firewall does not expand.

To illustrate the firewall concept, return to Figure 2. The four modules C_1 - C_4 are given as modified. We must rerun integration tests U_2 - C_1 , U_2 - C_3 , C_2 - U_4 , C_3 - U_7 , C_3 - U_8 , C_4 - U_5 , and C_4 - U_6 , but no unit tests for unchanged modules need be rerun. Figure 2 shows the firewall as bold arrows that separate the affected modules from the rest of the call graph—just as the firewall does in real testing.

I must make a final remark about our assumption that no errors exist in the system other than those due to the modified modules and that all unit and integration tests are reliable. Of course these assumptions do not hold true in practice, and thus our precise conclusions are no longer valid. However, we can make these conclusions practical by observing that testing the modules within the firewall is a sensible use of testing resources, even if we cannot

rule out errors existing elsewhere.

Computational experience. We also conducted an experiment to evaluate these regression testing concepts.² We used a student database program with 20 distinct modules, 32 modules, seven software features (major software functions in the specifications), and over 550 executable lines of Pascal code. The author provided four real modifications that we could evaluate with regression testing using reduced tests.

Table 1 shows the results of this study. Modifications 1, 2, and 4 show considerable reductions in the number of required tests, but modification 3 shows little reduction. The top four lines in Table 1 show the reason for this. Modification 3 has a slightly higher number of affected modules or module interactions than the other modifications. The biggest difference is in the number of affected features. Since modification 3 affects all features, the number of system tests does not decrease. Note that the design was good

Table 1. Regression testing evaluation.

	Modification			
	1	2	3	4
Affected source lines	25	80	23	57
Affected modules	2	4	8	8
Affected module interactions	2	3	16	10
Affected features	1	1	7	1
Regression tests				
Unit	15	22	50	40
Integration	32	32	120	80
System	46	24	130	38
Total tests				
Unit	27	40	67	66
Integration	246	278	275	307
System	106	130	130	158
Total tests	379	448	472	531
Regression tests	93	78	300	158
Percentage	24%	17%	63%	30%



1992 Gordon Bell Prize

For Outstanding Achievements in the Application of Parallel Processing to Scientific and Engineering Problems

Entries are due **May 1, 1992**, with finalists to be announced by June 30 and winners announced at the Supercomputing 92 conference in November 1992. Prizes of \$1,000 each will be awarded in two of three categories:

- Performance, based on megaflop rate on a machine with known performance compared against similar applications. If this is not possible, entrants should document their performance claims.
- Price/performance, based on performance divided by the cost of the smallest practical computational engine, including critical peripherals. Performance measurements will be evaluated as for the performance category.
- Compiler parallelization, based on the most speedup, measured by dividing the wall-clock time of the parallel run by that of a good serial implementation of the same job.

General conditions include demonstrating the utility of the program and machine. The judges will also consider how much the entry advances the state of the art in some field.

For more information or to enter, contact:

1992 Gordon Bell Prize
c/o Marilyn Potes
IEEE Computer Society
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264
Phone: (714) 821-8380

enough to anticipate modifications 1, 2, and 4 but was not maintainable for modification 3.

To establish the effectiveness of the test subsets and find which tests detected errors, we asked the programmer to introduce 13 logic errors. Of these, the tests detected 12, with the subsets as effective at detecting these errors as the full test sets. Of the 12 errors detected,

- unit testing detected eight errors,
- integration testing detected 11 errors, and
- system testing detected 12 errors.

Some lessons here mirror current practice. We could avoid bothering with unit or integration testing and conduct only system testing, but this would be very expensive and time-consuming. The values of both unit testing and integration testing are clear. Integration testing detects different errors than does unit testing. Developers should not only perform both types of testing but also save the data to do regression testing at these two levels.

Global variables. We also studied regression testing global variables⁵ and found that the global variables may be treated as parameters passed between modules for the purpose of regression testing. This is despite the fact that global variables are insidious in that many modules may become dependent through extensive use of global variables. A change in one or a few modules may affect change in modules throughout the entire system.

We are completing research on how developers should test global variables. The study is complex, but should provide an approach for developers to actually test the effects of global variables they insist on using.

References

1. H.K.N. Leung and L. White, "Insights into Regression Testing," *Proc. Conf. Software*

Maintenance, IEEE CS Press, Los Alamitos, Calif., Oct. 1989, pp. 60-69.

2. H.K.N. Leung and L. White, "A Study of Integration Testing and Software Regression at the Integration Level," *Proc. Conf. Software Maintenance*, IEEE CS Press, Nov. 1990, pp. 290-301.
3. M. Weiser, "Programmers Use Slices When Debugging," *Comm. of ACM*, Vol. 25, July 1982, pp. 446-452.
4. B. Korel and J. Laski, "Dynamic Program Slicing," *Information Processing Letters*, Oct. 1988, pp. 155-163.
5. H.K.N. Leung and L. White, "Insights into Testing and Regression Testing Global Variables," *Software Maintenance: Research and Practice*, John Wiley & Sons, Sussex, UK, Vol. 2, 1990, pp 209-222.

Lee White chairs the Department of Computer Engineering and Science at Case Western Reserve University in Cleveland, Ohio. His research interests include software testing, verification, and algorithm analysis.

White earned a BSEE from the University of Cincinnati and MS and PhD degrees in electrical engineering from the University of Michigan.

Hareton K.N. Leung works for Bell-Northern Research while he is completing his PhD work in computing science at the University of Alberta. His interests include test strategy development, process improvement, quality assurance, and software reliability engineering.

Leung earned a BS in physics and astronomy from the University of British Columbia and an MS in computing science from Simon Fraser University.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 192 Medium 193 High 194

IEEE MICRO

Editorial Calendar

JUNE 1992

Associative memories and processors

- Late-breaking developments in wide-word content-addressed memories (CAMs)
- Dynamic CAPP (parallel processor) architectures
- Fault-tolerant architectures for crucial control systems such as those in trains, space systems, etc.

Ad closing date: May 1

DECEMBER 1992

Special signal processors

- Recent information from the vital area of digital signal processors
- News about other techniques for signal processing
- Mixed analog/digital processors solve a real need
- Neural networks process signals following methods used by the human brain

Ad closing date: November 1

AUGUST 1992

European industry

- Recent developments in integrated circuit and microsystem technology from major European manufacturers

Ad closing date: July 1

FEBRUARY 1993

Automotive/traffic microelectronics

- Worldwide developments in microelectronics for traffic and driving assistance
- Latest developments in the European Prometheus and US IVHS programs
- News from Japan too
- Improving traffic safety with electronics

Ad closing date: January 2

OCTOBER 1992

Processing hardware for video communication

- ICs for HDTV compression
- ICs for HDTV communication
- Parallel processing systems with real-time video compression capabilities
- Multimedia systems with real-time video compression capabilities

Ad closing date: September 1

APRIL 1993

Advanced packaging/interconnect technology

- Critical trends and issues
- Flexible, glass, or diamond substrates
- Few-chip or 3D packaging
- Attachment, bonding, and connection technologies including fine-pitch surface mount, laser applications, known-good die, and interconnection trade-off analysis

Ad closing date: March 1

IEEE Micro helps designers and users of microprocessor and microcomputer systems explore the latest technologies to achieve business and research objectives. Feature articles in IEEE Micro reflect original works relating to the design, performance, or application of microprocessors and microcomputers. All manuscripts are subject to a peer-review process consistent with professional-level technical publications. IEEE Micro is a bimonthly publication of the IEEE Computer Society.

Advertising information: Contact Marian Tibayan, Advertising Department, IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax (714) 821-4010.

Articles may change. Please contact editor to confirm.



Send information for inclusion in *Micro News* one month before cover date to
Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

Second-generation RISC chips

Ware Myers, Contributing Editor

Authors presented papers describing 10 second-generation RISC processors at Compcon Spring 92, in February. Table 1 lists the papers, which are contained in the Digest of Papers, available from the Computer Society.

Most of these papers address new or unusual design features, so they do not contain complete application data. The most advanced chips have line widths of 0.75 to 0.80 μm and contain more than one million transistors. Most of them achieve high performance by using both superscalar and superpipeline techniques. With a million plus transistors the chip can, of course, contain many performance-enhancing features.

DEC Alpha. The Alpha is not merely a new chip. It is a new architecture that DEC intends "to withstand the test of time" for the next 25 years. EV4 is the first implementation of this architecture. DEC envisions more powerful implementations in coming years. "Future generations will be able to deliver up to a 1,000-fold increase in performance," the company said in its announcement.

The first implementation of Alpha is in 0.75- μm , 3.3V, CMOS technology. It contains 1.68 million transistors on a 1.68 \times 1.39-cm chip with 431 pins and operates at 200 MHz. The 64-bit CPU can issue two instructions each clock cycle to two of the four pipelined functional units: integer, floating point, branch, and load/store. Thus, the peak issue rate can reach 400 MIPS.

Alpha's architecture accommodates Digital's Open Advantage by supporting both OSF/1 and Open VMS operating systems. Thus, it provides an upgrade path from DEC's existing VAX architecture and an opportunity for any organization using Unix. DEC plans to license it to anyone.

Alpha can be employed as a single CPU in

personal workstations, or in large aggregations it can form massively parallel systems. Cray Research plans to use it in this way.

HP reduces path length. "The path length of a computation is the number of instructions needed to process the computation," Ruby Lee of Hewlett-Packard's Cupertino, California, facility, pointed out in the session devoted to HP's PA (precision architecture) RISC architecture. "The execution time of a computation is the product of the path length in instructions executed, the average cycles taken per instruction, and the cycle time of the processor."

It follows then that we can improve the performance of a processor if we can reduce one or more of these variables. Cycle time depends largely on the underlying technology. In the case of HP's current implementation, clock time is down to less than 10 ns. Superscalar and superpipelining reduce the cycles per instruction. RISC architecture originally speeded up processing by limiting the instruction set to relatively fast-executing instructions, permitting clock time to be reduced.

According to Lee, "The PA-RISC approach is the first to specify a datapath to meet minimum requirements, then to find opportunities to exercise multiple independent functional units with a single instruction so as to minimize path lengths and improve cost performance."

This opportunity is found in "multi-op" instructions that combine two or three operations in the 32-bit instruction after the fashion of VLIW (very long instruction word) architecture. A single instruction, thus, can initiate several operations on separate hardware resources in parallel. In effect, this technique manages to do more work in a single instruction, shortening the path length of instructions implemented in this way.

First, the design team measured the frequency

Table 1. Papers on recent RISC microprocessors, as contained in the *Digest of Papers, Compcon Spring 92*.

Processor	Company	Lead author
PA-RISC 1.1	Hewlett-Packard	Eric DeLano
Supersparc	Sun Microsystems	Greg Blanck
	Texas Instruments	
Pinnacle-1 (Sparc-compatible)	Cypress/Ross Technologies	Raju Vegesna
88110*	Motorola	Keith Diefendorff
Alpha (EV4)**	Digital Equipment	Richard Sites
NVAX**	Digital Equipment	Mike Uhler
PowerPC **	Apple/IBM/Motorola	Ron Hochsprung
i960 Cx	Intel	Elliot Garbus
Am29030/35	Advanced Micro Devices	Scott McMahon
LR33020 X-terminal controller	LSI Logic	Robert Tobias

*See page 40 in this issue.
 ** Extension of IBM RS6000 architecture

of occurrence of pairs or triples of operations. Then it selected several dozen to implement. "None of these multi-op instructions are difficult to generate from a compiler's point of view, since they map naturally to generic programming constructs," Lee noted. "The additional hardware required for these multi-op instructions is minimal compared to the additional performance provided."

Typically one of these multi-op PA-RISC instructions replaces two or three single-op instructions. In a few instances the replacement rate was in the range of five to 10 single-op instructions. "Overall, systems based on the PA-RISC architecture have achieved extremely competitive performance on both technical and commercial benchmarks," Lee concluded.

Metrology report

Measurement technology is the key to boosting semiconductor productivity, according to a US Department of Commerce report. *Metrology for the Semiconductor Industry* suggests that advances in metrology lead to breakthroughs in semiconductor technology.

Manufacturers who are better able to detect defective chips—and to prevent defects from occurring—develop more efficient processes. Furthermore, as designers build smaller chips and incorporate an increasing number of transistors, the margin for error in manufacturing shrinks.

The federal report cites several sources contributing metrology technology that can help manufacturers keep pace with microprocessor research, including federal agencies, universities, corporations, and cooperative research groups. A free copy of the report is available from Jane Walters, B3444 Technology Building, National Institute of Standards and Technology, Gaithersburg, MD 20899.

Germany recycles old machines

A new law will require German manufacturers to take back old electronic equipment, beginning in 1994. The Electronic Waste Order aims at reducing the 800,000 metric tons of electronic waste that reach the country's incinerators and dump sites each year. Some manufacturers already take back worn-out equipment and dismantle it

Micro bits

Computer manufacturer **Silicon Graphics** will acquire **Mips Computer Systems**. Some analysts see the move as an attempt to keep the Mips architecture competitive in the race to control the brains of the next generation of personal computers. Meanwhile, **Intel**, whose 386 and 486 microprocessors form the basis for the current generation of personal computers, announced it had signed a letter of intent to share technology with **VLSI Technology**.

Cray Research and **Sun Microsystems** will share hardware and software technology to create a seamless environment for Sun's systems and Cray's supercomputers. Cray recently formed a subsidiary, Cray Research Superservers, to make and sell Sparc products and joined Sparc International, the consortium to promote scalable processor architecture. The supercomputer innovator plans to introduce a massively parallel system next year with a peak performance of 100 Gflops.

The neural networks in **Janus** translate spoken sentences into or from English, German, or Japanese. Carnegie Mellon, Siemens AG, ATR of Kyoto, and the University of Karlsruhe collaborated to build the 400-word, continuous speech system.

The University of New Mexico distributes **Khoros**, a software development environment for information processing and data visualization, at no charge through file transfer protocol sites. The distributed computing system, runs on Sun, DEC, IBM, Hewlett-Packard, Next, Mips, and Cray machines and is accessible on e-mail at pprg.eece.unm.edu.

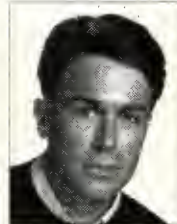
Three join editorial board

Editor-in-Chief Dante Del Corso has appointed three new members to the editorial board of *IEEE Micro*.

Teresa H. Meng is an assistant professor of electrical engineering at Stanford University. She will review manuscripts for the magazine. Meng earned a BS degree in electrical engineering at National Taiwan University and MS and PhD degrees in electrical engineering and computer science at the University of California, Berkeley.



Gilles Privat, a research engineer with France Telecom, the National Center for the Study of Telecommunications in Grenoble, will also review manuscripts.



He heads a research group investigating areas of parallel algorithms and VLSI architecture for image processing. Privat earned engineering and doctoral degrees

in signal and systems theory at Telecom Paris University.

Arun K. Sood is a professor of computer science at George Mason University. He will oversee plans for a new

department that will feature short technical notes and "dream chips" submitted by readers. Sood received a bachelor's degree from the Indian



Institute of Technology in Delhi and MS and PhD degrees from Carnegie Mellon University, all in electrical engineering. He is a

member of the IEEE Systems, Man, and Cybernetics Society's Administrative Committee and recently guest edited *IEEE Micro*'s theme issue on database machines.

to reuse parts. The new order requires them to recycle as much metal and plastic as possible. Precious metals will be salvaged and reused; other metals may be resmelted and used as slag in, for example, road paving.

Multimedia authoring program

A Stanford University programmer has developed software that lets students and professors create their own multimedia presentations with archive and original materials. From a Unix workstation, students can combine video and music with their own recorded commentary and typed-in text.

George Drapeau of the Academic Software Development Group of Stanford's Libraries and Information Resources created the program called Maestro (multimedia authoring environment). His prototype workstation includes a Sparcstation 2, microphone, laserdisc player, CD-ROM player for music and data, and stereo speakers.

The mouse-driven, icon-based interface allows users to access literary works available to on-line users of the

university's networks. A video editor lets users choose segments, add music, record voice commentary, and type in text and captions. A time line editor designates how segments overlap.

Drapeau says the program is not designed to produce professional presentations, but to create a simple tool that lets users concentrate on the task and not the computer. He offers the program as "freeware" to students.

Current literature

The Glossary of Computer Security Technology defines security terms used by US federal departments and agencies. The glossary provides multiple definitions, reflecting various uses by different federal agencies. Technical information on the 176-page publication is available from Edward Roback at (301) 975-3696.

National Technical Information Service, Springfield, VA 22161; \$26 (hard copy), \$12.50 (microfiche).

The five-volume ninth edition of the *Index and Directory of Industry Standards* lists 113,000 standards from 380 organizations. Twelve-thousand stan-

dards are revised since the last edition; 8,000 are new. The directory cites standards by subject, society/numeric, society, and ANSI concordance. Volumes 1 and 2 comprise the US set, volumes 3, 4, and 5 are the international set.

Global Engineering Documents, PO Box 19539, Irvine, CA 92713-9539; \$376 (complete set), \$195 (US volumes only), \$275 (international volumes).

The *Fall 1991 IEEE Standards Catalog Update* lists electrotechnology standards published since the institute issued its most recent catalog last September.

IEEE Standards, PO Box 1331, Piscataway, NJ 08855-1331; free.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 198 Medium 199 High 200

New Products

Send announcements of new microcomputer and microprocessor products to
Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264

Joe Hootman

University of
North Dakota

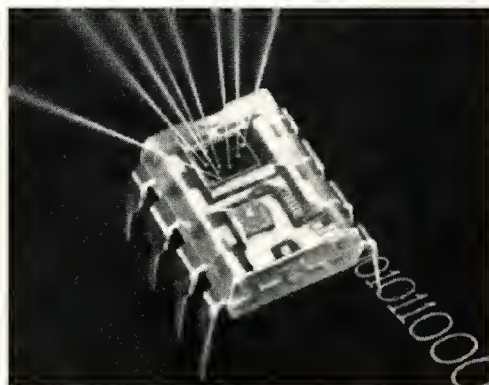
Devices and components

Light to frequency

For precision light measurements, the TSL220 converts light to digital signals. It comprises a photodiode and BiMOS current-to-frequency converter and connects directly to a microprocessor or a digital control circuit. The CMOS-compatible output voltage is a pulse train with a frequency directly proportional to light intensity of the diode.

The device features a dynamic range of 118 db and output levels of over 100 KHz in office desk lighting and as low as 1 Hz in the dark. The converter functions with a 5 to 10V power supply and in temperature ranges of -25° to 70°C. It is housed in an eight-pin, clear DIP package. *Texas Instruments; \$4.61 (1,000s).*

Reader Service No. 10



Texas Instruments' TSL220

1-Mbit SRAM

PSM44039 is a processor-specific memory (PSM) chip that works as a secondary cache for high-end, synchronous RISC processors. The 1,179,648-bit chip is a self-timed, synchronous, CMOS SRAM organized as 128 Kwords by 9 bits. Available cycle times range from 15 to 25 ns.

The PSM chip operates from a 5V supply in temperatures of 0° to 70°C. *Paradigm; \$85 (20-ns version, 1,000s).*

Reader Service No. 11

10Base-T interface

The SN75LBC086 differential driver/receiver is a one-channel interface for concentrators, repeaters, and bridges in twisted-pair Ethernet systems. The 24-pin, 300-mil device features a squelch circuit for noise immunity beyond 10Base-T standard requirements. As also required by the standard, it provides jabber control, collision detection, signal quality, and link test functions. A loopback mode permits testing the data path while still connected to the network. *Texas Instruments; \$8.40 (1,000s).*

Reader Service No. 12

One-chip display driver

For vacuum-fluorescent displays, the M66004 controller/driver generates 16 characters from RAM and 160 characters from ROM, for a variable display length up to 16 display digits in 5 × 7 segments. Two built-in static points drive LEDs or control peripheral ICs.

A three-line serial bus to the microcontroller receives data without needing a buffer. The chip also features an eight-step dimmer control, cursor display, and two scan-cycle formats. *Mitsubishi; \$5.12 (1,000s).*

Reader Service No. 13

Pulse-width modulators

Designers can build 500-KHz, off-line power supplies and DC-to-DC converter applications with a line of pulse-width modulators. The LT1241-1245 modulators feature temperature-compensated reference, high-gain error amplifier, current-sensing comparator, 50-ns current sense delay, start-up current of less than 250 µA,

Neuron chip

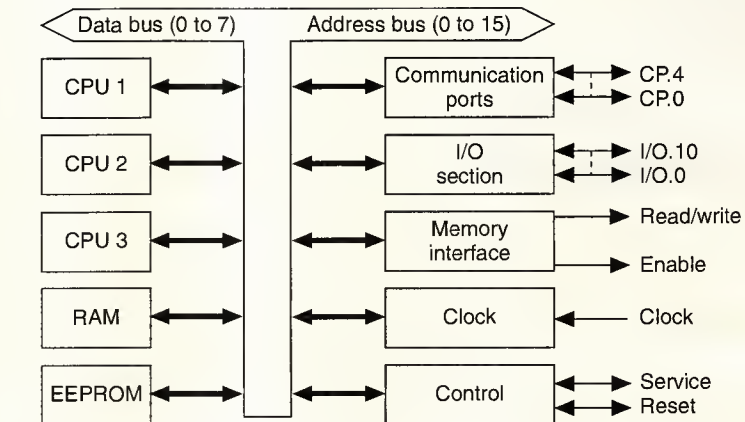
David Sims, Assistant Editor

Motorola's MC143150 Neuron Chip is a microcontroller with an embedded protocol that forms the heart of remote nodes in networks based on Echelon's Lontalk. Lontalk is a distributed sense and control network protocol for industrial, commercial, and residential systems that is specially designed to transfer small amounts of data, and thus reduce costs.

Because sense and control networks transfer relatively small packets of data compared to other network systems (for example, "Turn down the heat in the board room!" instead of "Here is the report on last quarter's production..."), they can get by with slower data transfer rates. Lontalk sends data packets of 15 bytes at up to 1.25 Mbps, considerably slower than Ethernet's 10 Mbps or some of the newer systems transmitting up to 150 Mbps.

According to Al Mouton, strategic planning manager in Motorola's Lonworks Products organization, the slower transfer rate is one way to reduce the costs of these systems. Another is the on-chip incorporation of all the functions needed to process inputs from sensors and control devices and respond with commands. Each Neuron Chip acts as an "intelligent controller" capable of continuing its monitor and command functions even if the network gets disconnected.

Mouton compared the difference between Lontalk and a centrally



Motorola's MC143150 Neuron Chip

based system to the difference between desktop PCs and a mainframe system with "dumb" terminals.

"If a central computer system goes down, everyone just sits there staring at blank screens," he said. "With PCs on every desk, if the network fails, you can go on working. You just can't transfer data."

On-chip features include

- three 8-bit pipelined processors,
- an 11-pin I/O port programmable in 24 nodes,
- two 16-bit timer/counters,
- a five-pin communications port to support network transceivers,
- a 2-Kbyte SRAM,
- 512 bytes of EEPROM with charge pump,
- an external memory interface,
- a sleep mode, and
- a 48-bit ID number unique to each device.

Mouton also said that the on-chip incorporation of a protocol reduces the cost of each node. Echelon and Motorola hope that success for Lontalk will make it a de facto standard for local network command systems. Nodes within other systems that include the software, systems, and components to form a complete node can cost up to \$50 per unit. Motorola expects, with volume production, to reduce the cost of the Neuron Chip to under \$5 by 1994.

Given the unlimited number of nodes acceptable within Lontalk's hierarchical architecture, Mouton sees a wide range of potential applications, from manufacturing lines to home automation, from building security to systems in a recreational vehicle.

The Neuron Chip is packaged in a 64-pin quad flat pack. Motorola; \$11.78 (1,000s).

Reader Service No. 14

and a high-current totem pole output stage suited to drive power MOSFETs. The chips incorporate blanking in the current sense comparator to prevent the leading edge current spike from prematurely tripping the comparator. *Linear Technology; \$3.74 (1,000s).*

Reader Service No. 15

Software

Utilities boost DOS machines

PC-Kwik Power Pak's utilities speed the performance of 286-, 386-, and 486-based machines by employing a disk cache, screen and keyboard accelerators, and a print spooler. A data buffer

in RAM boosts application speed. Accelerators speed cursor movement up to 126 cps and scrolling up to three times faster than standard. The print spooler stores data for the printer so the system can return to an application. A disk cache shares memory between PC-Kwik utilities and lends

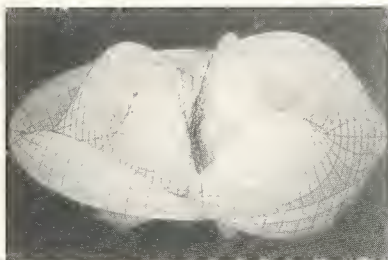
memory to application programs as needed. Other utilities in the set include a screen blanker that operates on a timer or with a hot key, and a command-line editor. *Multisoft*; \$70.

Reader Service No. 16

Algebra system

Maple V for Amiga DOS is an interactive computer algebra system that delivers 3D postscript and image file format output graphics. Its mathematics library includes more than 2,000 functions, supported by an Arexx port. The program supports Commodore Amiga 1000, 2000, 2500, and 3000 on an Amiga DOS version 2.0 or higher with 2 Mbytes of RAM and 8 Mbytes free disk space. *Waterloo Maple Software*; \$450.

Reader Service No. 17



Maple V sample output

Hspice graphic interface

Graphical Simulation Interface is a point-and-click, mouse-driven graphical user interface that provides interactive capabilities for quick analysis of Hspice simulations in an X-Windows environment. A machine-independent file format connects Hspice to GSI so users can run Hspice on a mainframe and view results graphically on a workstation.

GSI also provides concurrent simulation and wave review, point-and-click node property and selection, interactive curve measurement, automatic storage of last curve display, and flexible viewing with zoom, pan, multiple panels, and multiple simulation data. GSI supports most Unix X-Windows workstations. *Meta-Software*; \$2,000.

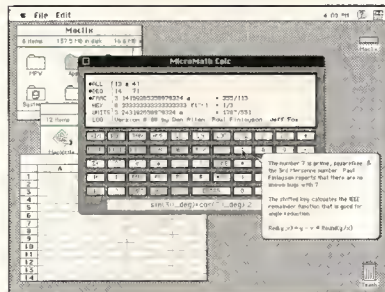
Reader Service No. 18

Scientific calculator for Mac

Micro Math Calc, a desk accessory calculator for the Macintosh, offers advanced features for programmers, mathematicians, and engineers. Real, complex, and Gaussian numbers can carry information on associated units and systems.

Users can enter and view numbers in binary, octal, decimal, hexadecimal, and with their corresponding ASCII characters. The calculator supports shifting operations, integer division, and logical bitwise operations such as Or, Not, And, and Xor. A bits function lets users quickly determine binary quantities. The calculator complies with the Standard Apple Numeric Environments and IEEE standards. Available for System 7, Micro Math Calc requires 100 Kbytes of memory. *Micro Math Scientific Software*; \$99.

Reader Service No. 19



Micro Math Calc

Terminal emulators

KEAterm 420, version 2 emulates the DEC VT420 terminal for DOS machines running Windows. Emulated functions include multiple sessions and pages, double high/wide characters, and 132-column support. Extensions include user-definable keyboard mapping and attribute-mapped colors.

The software supports IBM's enhanced keyboard, DEC's LK250, and KEA's Power Station keyboard. Pull-down menus display in English, French, or German. Version 2 enhancements include network/file transfers and script language enhancements. Other features in this latest upgrade include interfaces for TCP/IP, KEALink TCP, and Super

Kermit file transfer protocols.

A second product, KEAterm 340, includes all the capabilities of KEAterm 420 and supports Regis, Tektronix, and sixel graphics. *KEA Systems*; \$245 (420), \$395 (340).

Reader Service No. 20

Signal processing hardware and software

20-MHz signal processor

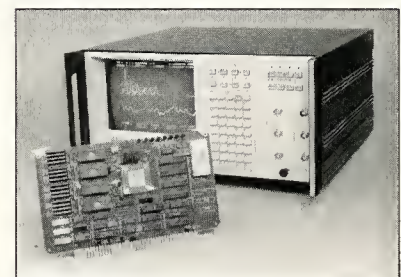
The 20-MHz ADSP-21020 floating-point DSP cycles in 50 ns and calculates a 1,024-point FFT in 0.96 ms. The manufacturer says its architecture, optimized for signal processing applications, suits it for image processing, graphics, radar and sonar, speech recognition, and advanced audio applications. The chip comes in commercial (0° to +85°C) and military (-55° to +125°C) temperature ranges, in a 223-lead pin grid array. *Analog Devices*; \$198 (1,000s).

Reader Service No. 21

DSP speeds waveform analysis

Model 683 is a DSP add-on board that extends Analogic's Model 6100 Waveform Analyzer by a factor greater than 300. According to the company, the add-on board's 25-Mflops processor computes and displays an 8K-point FFT in milliseconds and a 16K × 16K cross-correlation analysis in one second. The speed allows real-time signal processing and spectral analysis. Model 683 uses a 32-bit floating-point DSP slaved to Model 6100's CPU. *Analogic*; \$2,995.

Reader Service No. 22



Analogic's Model 683 and Model 6100

Smooths data

Data Smoother processes the scattered points of original data and creates a waveform while preserving any abrupt change in values. Release 2.0 for MS-DOS enables users to enter data from the keyboard or other ASCII sources. The program process 1,500 points of data in seconds and displays original and smoothed data in tables, alphanumeric strip chart, or graphically. Users can choose from predefined labels or create their own and save on disk. *Dynacom*; \$50.

Reader Service No. 23

16-bit input module

The SBX-416 is an isolated, 16-bit analog input module that uses a successive approximation analog-to-digital converter to process at up to 20 KHz. A software driver, compatible with Microsoft and Borland C compilers, generates a serial data stream. An optically isolated external trigger initiates data conversion. The module self-calibrates on start-up. *Systek: \$559 (100s).*

Reader Service No. 24

Asynchronous servers

Asynserv2 and Asynserv8 (two- and eight-port units) allow LAN users to share modems; remote users can access the LAN and locally process under remote control. The two asynchronous communication servers support dial-in and dial-out communications at up to 57.6 Kbps per port, allowing 14.4-Kbps V.32bis modems with V.42bis data compression to run at full speed.

The systems include hardware and all necessary software to work with IPX or Net BIOS LANs. Other features include call-back security, host keyboard locking, screen blanking, multiple file transfer capabilities, script language, dialing directory, mail, chat mode, and pop-up menus. Asynserv measures 38.3 cm \times 8.0 cm \times 25 cm and feeds off a universal power supply (90 to 260V). *MNC International*; \$2,495 (*Asynserv2*). \$3,895 (*Asynserv8*).

Reader Service No. 25

Windows software

Windows development tool

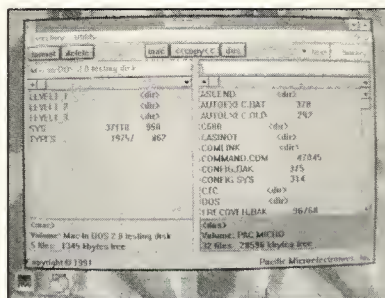
Desktop users in multiuser networks and client-server environments can develop applications in a Windows environment with Open Insight. It lets developers create database applications or link to SQL Server, Oracle, or other systems to create client-server applications. Open Insight includes development tools and an active data dictionary that gives users an integrated view of data sources, including dBase, ANSI SQL, SQL Server, ASCII, and DB2. Quarterly updates, add-on utilities, and a year of telephone technical support are included. *Revelation Technologies*; \$895.

Reader Service No. 26

Mac-in-DOS for Windows

Mac-in-DOS, one of several programs that copy and convert files between Macintosh and DOS formats, is now available in a Windows format. Version 2.0 lets users run the program with Microsoft Windows 3.0. Function keys perform the main DOS file-handling functions, including changing subdirectories and deleting or copying files. *Pacific Micro*; \$99.

Reader Service No. 27



Pacific Micro's Mac-in-Dos

DOS or Windows applications

Developers can build Windows or DOS applications with APL Plus II/386, version 4. The program creates APL applications with graphical user interface for the Microsoft Windows 3.0 environment. Version 4 also interfaces to non-APL software, including most DOS

software with an application programming interface for C programmers. It also includes an interface to Borland's Paradox Database Manager, a screen interface toolkit, and Super VGA (800 × 600) 256-color graphics support. *STSC; \$1,700, \$495 (upgrades).*

Reader Service No. 28

Mouse-driven help systems

Two windows programs let users create help systems or data validation entry screens for Microsoft Windows without writing code, by pointing and clicking with a mouse. Robo Help includes a tool palette that can simplify construction of help systems. It generates source codes for indexes, categories, defined terms, and hypertext files.

Magic Fields lets users develop data compilation fields by pointing and clicking to predefined data entry fields, and adding custom-designed fields. Users can specify fonts, colors, and a grayed 3D effect. It includes a library of numeric, text, alphanumeric, and monetary objects. *Blue Sky Software; \$495 (Robo Help), \$349 (Magic Fields).*

Reader Service No. 29

Document management

DOS users can retrieve or scan word processing, database, spreadsheet, or graphics files with DE/Cartes Document Manager. The icon-based system stores files with names up to 64 characters long. Each document can have an unlimited number of revisions online. Users can define their own hierarchy of storage.

One mouse click selects a file, a second accesses a user-defined note, a third loads the program. On remote systems, access is restricted to one user at a time per document, and unauthorized users can be locked out. DE/Cartes requires a DOS machine, MS-DOS 3.0, Microsoft Windows 3.0, graphics card, mouse, 5 Mbytes of hard disk space, and 2 Mbytes of RAM. *Desktop Engineering International*; from \$147.50 (introductory price).

Reader Service No. 30

Display and scan peripherals

Video array with interface

Media Wall, an array of monitors with a computer interface, is a multimedia presentation system integrating computer animation, graphics, and text with full motion and still video. It includes an adapter card for the Macintosh, a satellite control unit containing special-effects hardware, and an array of stackable video monitors or projectors.

Users can display duplicate or different images in a variety of modes. Or they can display one high resolution (3,200 × 2,400) image tiled across the array. Monitors align up to 27 in a line or circle, or in a grid pattern up to five by five. *RGB Spectrum*; \$26,000 (interface board and controller box), \$40,000 (complete unit with nine monitors). *Macintosh not included.*

Reader Service No. 31

Pen base for desktops

Displaypad connects to desktop computers to make a pen-based system. Information written or drawn on the tablet simultaneously appears on the monitor. A cordless stylus senses tip pressure, height, and angle. The stylus' resolution is 1,270 lines per inch, and it has a data output of 200 points/s. The tablet features 640 × 480 resolution, 64 shades of gray, and a flush surface that allows smooth pen operation.

To install, a display card replaces the existing VGA card. Displaypad works with DOS, Macintosh, and Sun systems. *Cal Comp*; \$2,500.

Reader Service No. 32



Cal Comp's Displaypad

Faster rasters

Two raster plotters for Macintosh applications achieve what the manufacturer says are the fastest speeds in their class. Model 2400, with a 24"-wide, D-size format, plots at 4" per second and can format and plot a D-size plot in under 40 seconds. Model 3600, with a 36"-wide, E-size format, outputs 2" per second. The 200-dpi, monochrome plotters use Microspot Mac Plot DMA driver software and a NuBus interface card to process output from Claris CAD, Mac Project, Pixel Paint, Super Paint, Power Point, Freehand, Dreams, and other Quickdraw programs. *Atlantek*; \$12,500 (2400), \$14,500 (3600).

Reader Service No. 33



Atlantek's Model 3600

Prints 30 pages per minute

The LC-7030 nonimpact printer outputs 30 pages per minute. Two disk drives hold fonts; ambitious users can replace one with a 52-Mbyte hard disk for more storage. The controller supports HP PCL 5, Post Script, and DEC LN03 Plus emulations. Connects to a printer via RS-232 or RS-422 serial ports, Centronics parallel ports, Ethernet, TCP/IP, and twinaxial or coaxial cables. *Advanced Technologies International*; \$16,480 (simplex), \$21,430 (duplex).

Reader Service No. 34

In-house sign production

The Image Crafter creates multicolored graphic applications for signage and presentations. The package includes a desktop vinyl cutter/plotter, software, and a hand-held scanner. Users scan an image into their DOS machine (MS-DOS 3.1 or higher).

There, they can manipulate and clean up the image in a window-based program, before sending the finished product to the plotter. *Kroy Sign Systems*; \$2,195.

Reader Service No. 35



Kroy's Image Crafter

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 189 Medium 190 High 191

Product Summary

Joe Hootman

University of North Dakota

Manufacturer	Model	Comments	R.S.#
Boards			
Brooktrout Technology	TR112 fax card	PC/AT-compatible Twin Channel board contains two transceivers for multichannel facsimile applications. By taking advantage of direct-dialing services, an autorouting version sends incoming messages automatically to LAN fax-mail server users. \$1,995 and \$2,495 (autorouter). Quantity discounts available.	80
Rohm Corporation	Memory cards	Credit card-size and smaller SRAM, DRAM, mask ROM, one-time PROM, and flash-memory cards support applications in which small COB solid-state units can replace floppy disks. The 32-Kbyte to 6-Mbyte semicustom products promise 100-ns to 150-ns access times. From \$35 to \$950 (100s); 12 weeks ARO.	81
Star Tech	860 Edge add-in	With 32 to 128 Mbytes of DRAM and math/vector libraries, the Macintosh II i860 coprocessor accelerates the Pixar Mac Renderman photorealistic renderer, fitting into one Nubus slot. A developer's version supports the large floating-point operations required in scientific applications. \$8,000 (32 Mbytes).	82
Traquair Data Systems	HEPC2 parallel processor	TMS320C40-based, PC/AT-compatible board supports parallel, image, and digital signal processing, as well as graphics computations. Supporting up to three TIM-40 TMS daughter boards, HEPC2 provides up to 200 Mflops of floating-point performance (1.1 billion operations/s). From \$1,539 (mother board); from \$1,500 (TIM-40s).	83
Software			
Micro Touch Systems	Power Keypad	PC Unmouse software lets Microsoft Windows and DOS users control the cursor and execute macros by touching a keypad marked on a template inserted beneath the Unmouse glass surface. To click, users press down on the glass. Free upgrade to Unmouse owners.	84
Motorola	Smart Model for 68302	Behavioral-level simulation model for the 68302 integrated multiprotocol processor lets designers develop, debug, and optimize the hardware operation of 302-based designs before committing to physical prototypes. Smart Model features intelligent error-checking, user-defined timing, and VHDL interoperability. \$4,000 for one-time technical licensing fee, plus Logic Automation library license fee.	85
Systems			
Anorad	Anoguide AG-12, AG-14	Controller- and linear motor-equipped series boosts speeds to 100 ips and up to 120 lbs. of force at a 25-percent duty cycle. For noncontact operation in an industrial environment, the Anoline brushless, iron core coil assembly attaches to a moving	86

Manufacturer	Model	Comments	R.S.#
		slide while the stationary permanent magnets are mounted to the stage's base plate.	
Ariel	IRCAM workstation	Designed for compute-intensive applications on the Next Cube, this signal-processing workstation uses two i860 RISCs combined with a DSP56001 to perform at 160 Mflops and 93.5 MIPS. The CPOS/FTS real-time operating system provides protected multi-tasking kernel, memory management, file I/O, interprocessor communications, and process management facilities. \$14,995; university discounts available.	87
Neotronics/Laser Monitoring Systems	TE-TC temperature controller	Controller with built-in safety features supports semiconductor devices that need to be operated at lower than ambient temperatures. A four-digit, seven-segment LED shows either set or measured values of temperature, resistance, and current.	88
Nighthawk Electronics	DXS-16 data exchanger	Several computers can share a number of peripherals (printers, modems, file servers) with the 16-Mbyte DXS-16 as a LAN alternative. Each unit maintains 500,000-bps speeds on up to 16 serial, parallel, or 3270 coaxial/twin-axial ports.	89
PI Systems	Infolio pen computer	Pen-based, 2.9-lb., 9 × 10 × 1.2-in., PCMCIA-memory computing tool features a 640 × 480-pixel VGA-quality, reflective LCD. Infolio integrates a Cal Comp cordless stylus and Motorola 68331-based hardware with PDX-framework database software, a graphical user interface, and task-specific application software for mobile information collection and management solutions. \$1,895.	90
Miscellany			
I-Con Industries/Antel Corporation	Multiwire backplane	Futurebus+ backplane in A, B, and F profiles with 64- and 128-bit data widths comes in 5-, 9-, and 14-slot configurations. The multilayer board uses precision discrete wires rather than etched circuits for signal interconnection and eliminates signal layers and the number of corresponding etched voltage and ground planes required for impedance reference.	91
Multiaccess Computing	MCC-1000F adapter	Frame relay service adapter card provides Macintosh II users with multiple, presubscribed virtual connections across metropolitan or wide-area networks. The one-board controller occupies one Nubus I/O slot on the system board and runs at T1 or fractional T1 rates over a T1 facility. \$2,995; 60 days ARO.	92
Parsytec	TIP I/O bus boards	TIP series expands I/O on transputer-based parallel processing systems, transmitting data simultaneously, in parallel, to multiple transputer nodes via its broadcast function. The broadcast rate equals $n \times 100$ Mbytes/s, where n equals the number of receivers. Series includes the TIP-VPU/T8 T805 processor board, TIP-MFG monochrome frame grabber, and the TIP-CGD color graphic display board. From \$4,600 each; 3 or 4 weeks ARO.	93

FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

District Manager: D. Rodney Brooks; Tel: (415) 905-0260; Fax: (415) 896-1512.

Recruitment and Classified Advertising: D. Rodney Brooks; Tel: (415) 905-0260; Fax (415) 896-1512.

Director of Sales: Randall L. Stickrod, 544 Second St., Suite 200, San Francisco, CA 94107; Tel: (415) 905-0260; Fax: (415) 896-1512.

For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

IEEE MICRO, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

Coming in June

IEEE Micro's associative memories and processors issue features articles on:

- A dynamic associate processor for machine vision applications
- A module for a heterogeneous vision architecture
- A pattern-addressable memory
- And more...

**Also, look for On the Edge's
discussion of Object Encyclopedia
Technology standards.**

RS # Page #

Advanced Technologies Int'l	34	93
Analog Devices	21	91
Analogic	22	91
Anorad	86	94
Antel Corp.	91	95
Ariel	87	95
Atlantek	33	93
Blue Sky Software	29	92
Brooktrout Technology	80	94
Cal Comp	32	93
Desktop Engineering Int'l	30	92
Dynacomp	23	92
GIGATEC SA	1	C.IV
I-Con Industries	91	95
KEA Systems	20	91
Kroy Sign Systems	35	93
Linear Technology	15	90
Meta-Software	18	91
Micro Math Scientific Software	19	91
Micro Touch Systems	84	94
Mitsubishi	13	89
MNC International	25	92
Motorola	14, 85	90, 94
Multiaccess Computing	92	95
Multisoft	16	91
Neotronics/Laser Monitoring Systems	88	95
Nighthawk Electronics	89	95
Pacific Micro	27	92
Paradigm	11	89
Parsytec	93	95
PI Systems	90	95
Revelation Technologies	26	92
RGB Spectrum	31	93
Rohm Corp.	81	94
STSC	28	92
Star Tech	82	94
Systek	24	92
Texas Instruments	10, 12	89
Traquair Data Systems	83	94
Waterloo Maple Software	17	91

THE FOLLOWING INFORMATION IS AVAILABLE:

Contact the Publications Office; to facilitate handling, please request by number.

- Membership application, student #203, others #202
- Periodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Compmail electronic mail brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures #193
- Student scholarship information #192
- Volunteer leaders/staff directory #196
- IEEE senior member grade application #204 (requires ten years practice and significant performance in five of those ten)

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

PURPOSE

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

MEMBERSHIP

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.



IEEE COMPUTER SOCIETY®

A member society of the
Institute of Electrical and Electronics Engineers, Inc.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and product reviews.

Periodicals. The society publishes seven magazines and five research transactions. Refer to membership application or request information as noted above.

Conference Proceedings, Tutorial Texts, Standard Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. Over 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

OMBUDSMAN

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

EXECUTIVE COMMITTEE

President: Bruce D. Shriver*
17 Bettea Drive
Ossining, NY 10562
Phone: (914) 762-3251
Fax: (914) 941-9181

President-Elect: James H. Aylor*
Past President: Duncan H. Lawrie*

VP, Conferences and Tutorials: Barry W. Johnson (1st VP)*
VP, Educational Activities: Raymond E. Miller (2nd VP)*
VP, Membership Activities: Fiorenza C. Albert-Howard*
VP, Press Activities: Yale N. Patt*
VP, Publications: Harold S. Stone*
VP, Standards Activities: Gary Robinson†
VP, Technical Activities: Joseph Boykin†

Secretary: Ronald G. Hoelzeman*
Treasurer: Laurel V. Kaleda†
IEEE Division V Director: Bill D. Carroll†
IEEE Division VIII Director: Helen M. Wood†
Executive Director: T. Michael Elliott†

*voting member of the Board of Governors
†nonvoting member of the Board of Governors

BOARD OF GOVERNORS

Term Expiring 1992:

Alicja I. Ellis, Ronald G. Hoelzeman, Tadao Ichikawa,
C.V. Ramamoorthy, Sallie V. Sheppard,
Harold Stone, Akihiko Yamada

Term Expiring 1993:

Fiorenza Albert-Howard, Jon T. Butler, Michael C. Mulder,
Yale N. Patt, Anneliese von Mayrhauser,
Benjamin W. Wah, Ronald Waxman

Term Expiring 1994:

Mario R. Barbacci, Luis-Felipe Cabrera, Wolfgang K. Giloi,
Guylaine M. Pollock, John P. Riganati, Ronald D. Williams,
Thomas W. Williams

Next Board Meeting

June 5, 1992, 8:30 a.m.

Sheraton New Orleans, New Orleans, LA

SENIOR STAFF

Executive Director: T. Michael Elliott
Publisher: H. True Seaborn
Director, Conferences and Tutorials: Anne Marie Kelly
Director, Finance and Information Services: Tod S. Heisler
Director, Board and Administrative Services: Violet S. Doan
Assistant to the Executive Director: Sandra K. Plau

COMPUTER SOCIETY OFFICES

Headquarters Office

1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone: (202) 371-0101
Fax: (202) 728-9614

Publications Office

10662 Los Vaqueros Cir.
PO Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information:
(714) 821-8380
Publication Orders: (800) 272-6657
Fax: (714) 821-4010

European Office

13, Ave. de L'Aquilon
8-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05

Asia/Pacific Office

Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 3408-3118
Fax: 81 (3) 3408-3553



IEEE OFFICERS

President: Merrill W. Buckley, Jr.
President Elect: Martha Sloan
Past President: Eric E. Sumner
Secretary: Karsten E. Drangeid
Treasurer: Theodore W. Hissey, Jr.

VP, Educational Activities: Edward A. Parrish
VP, Professional Activities: Arvid G. Larson
VP, Publication Activities: James T. Cain
VP, Regional Activities: Luis T. Gandia
VP, Standards Activities: Marco W. Migliaro
VP, Technical Activities: Fernando Aldana

COMM NEXUS

Presents

The LiSBUS™ Async I/O System

A solution of the 1990's for today's data transmission problems.



Outstandingly Simple and Reliable because LiSBUS™ is based on a breakthrough technology which uses the impedance of the bus cable to replace binary addresses. Consequently, data transmission management is greatly simplified and much more reliable than today's equivalent systems which require expensive software, hardware, and personnel investments.

Outstandingly Practical because it is easy to install and operate. No special tools, workbench, or electronics expertise are needed. Anyone can be up and running in minutes. Just plug in the external modules and configure the system with the user-friendly LiSBUS™ Link Control Software. Each external module measures only around 2in. by 2in.

Outstandingly Flexible because a user can connect up to 60 peripherals or computers to a controlling computer through their RS-232C (COM) ports. To add peripherals, just extend the bus cable and add modules.

At an Unbeatable Price because at \$650* for the LiSBUS™ Starter Pack, no alternative offers all these advantages combined into one product without spending a much higher

amount. The **Starter Pack** includes all the user needs to connect four peripherals and a complete set of **LiSBUS™ Software Development Tools** to create custom applications.

LiSBUS™ Async I/O System: A product of our CommNexus™ line of communication systems. GIGATEC is committed to offering products and servicing its customers in the best tradition of Swiss quality. We provide our customers with:

- **Technical Support.** Registered buyers can obtain technical support from our qualified engineers.
- **Users and Developers Group.** Organized for encouraging software developments using the products of the CommNexus™ family.

For more information and ordering contact:

In the USA and Canada:

In Europe:

Toll Free (800) 945-3002

(excl. Hawaii)

Mon.-Fri. 9am - 9pm EST

GIGATEC (USA), Inc.
871 Islington Street
P.O. Box 4705
Portsmouth, NH 03802-4705 USA
Tel. (603) 433-2227
Fax (603) 433-5552

GIGATEC SA
Ch. des Plans-Praz
1337 Vallorbe SWITZERLAND
Tel. 41 21 843 37 36
Fax 41 21 843 33 25

* specifications and prices subject to change without prior notification
Visa and MasterCard/EuroCard accepted. - CommNexus™ and LiSBUS™ are trademarks of GIGATEC SA.

Reader Service Number 1